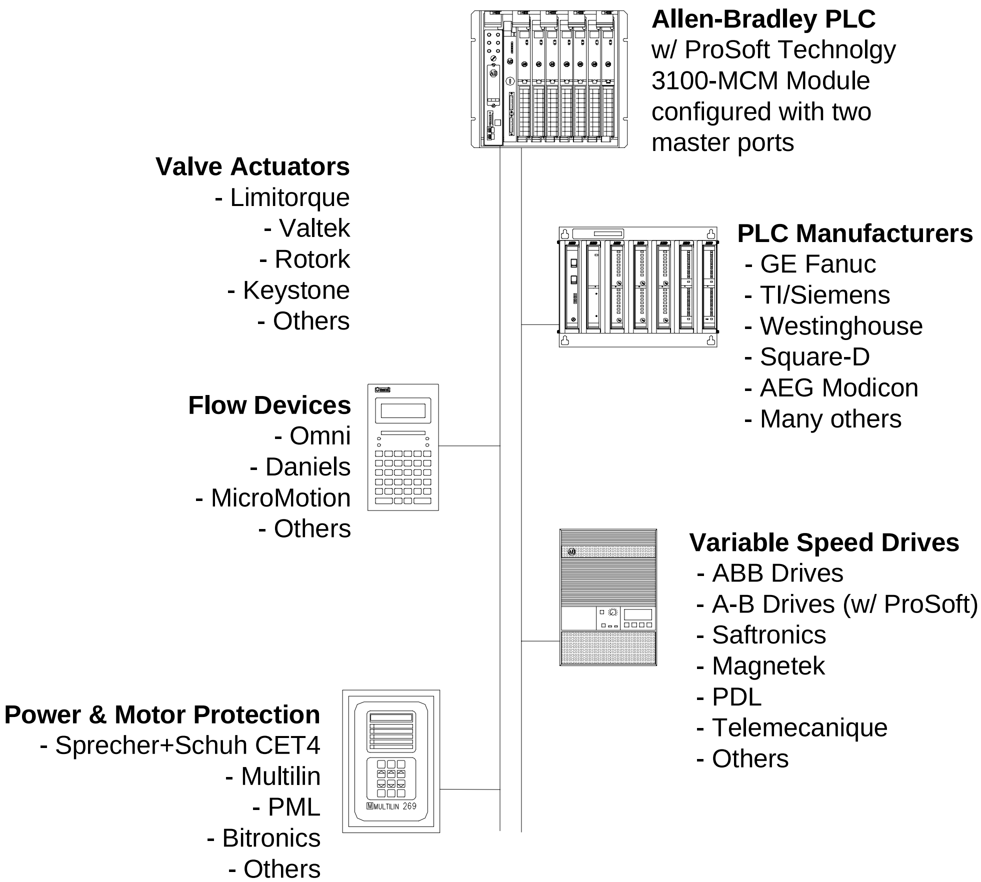
Prosoft_Modbus_3100_3150MCM_iom_D497

3100/3150-MCM Modbus Communications

Revision 2

April 1997



USER MANUAL

ProSoft Technology, Inc.
9801 Camino Media, Suite 105
Bakersfield, CA 93311
(661) 664-7208
(661) 664-7233 (fax)
E-mail address: prosoft@prosoft-technology.com
Web Site : http://www.prosoft-technology.com

Please Read This Notice

Successful application of the MCM card requires a reasonable working knowledge of the Allen-Bradley PLC or SLC hardware and the application in which the combination is to be used. For this reason, it is important that those responsible for implementing the MCM satisfy themselves that the combination will meet the needs of the application without exposing personnel or equipment to unsafe or inappropriate working conditions.

This manual is provided to assist the user. Every attempt has been made to assure that the information provided is accurate and a true reflection of the product's installation requirements. In order to assure a complete understanding of the operation of the product, the user should read all applicable Allen-Bradley documentation on the operation of the A-B hardware.

Under no conditions will ProSoft Technology, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of the MCM product.

Information in this manual is subject to change without notice and does not represent a commitment on the part of ProSoft Technology, Inc. Improvements and/or changes in this manual or the product may be made at any time. These changes will be made periodically to correct technical inaccuracies or typographical errors.

New Features in Revision 2

Revision 2 of the MCM product represents the first major upgrade to the product. Incorporated in the product are many new features, some which have been suggested by our existing customers and some as the result us learning how our customers are using the product. A description of some of these new features follows:

Modbus Master Driver

Command Status Bits

Added to the information returned to the ladder logic are command 'Done' and 'Error' bits. These bits allow the ladder logic to track each Modbus Master command's execution and its success or failure. In conjunction with the new Command Control Mode (see below), the ladder program is now able to control the execution of each command.

Command Polling Time

Each command can now be configured with a 'Polling Time' to effectively control the frequency of execution. The timer has a resolution of one(1) second and each command can be configured up to 65535 seconds.

Command Control Mode

The Command Control Mode is most likely the most requested feature we have been asked for. In this mode, the Modbus Master driver only executes commands in the command list upon receiving an 'Enable' bit from the ladder logic. The module returns 'Done' and 'Error' bits as part of the regular command status information returned to the ladder (See discussion above) allowing commands to be sent only once if desired.

Event Initiated Write Commands

The Modbus Master driver in the MCM module supports the execution of Event Driven Write Commands. This type of support provides the ladder logic program with another method beyond the Conditional Write Commands (documented in the manual) and the Command Control Mode to send Modbus Write commands to a slave (FC 5,6,15,16).

Modbus Slave Driver

Pass-through Mode

The Pass-Through Mode allows a Modbus Slave port to pass write commands received from a host directly across the backplane for handling by ladder logic. Although this feature requires more ladder logic in order to implement a solution, there are certain situations where this functionality can be useful. Some of these situations include:

1. When the slave needs to know when it has been written to
2. When the acceptance of data may require some conditioning
3. When the host's write data registers must overlap the read register space

Routing Mode

The Routing Mode allows a Modbus Slave port to 'route' commands received from a host to the master port. A list of up to six (6) routed slave addresses can be entered in the configuration table. Whenever a host command (either read or write) is received on the slave port which matches one of the six addresses, the command is routed out the master port and the response is routed back to the slave port.

Quick Start Implementation Guide

Integration of the MCM module into a PLC or SLC application is easier if a series of steps are followed. In order to assist the first time users of our products in getting operational quickly, we have come up with this step-by-step implementation guide.

First Time Users

Although the following steps are to assist you in implementing the module, we recommend that you attempt to experiment with the example logic provided on disk with the module or available off our FTP site before laying out your application. This step will allow you to gain insight into how the module works prior to making decisions which will impact the long term success of the installation.

Starting with one of the ladder logic programs provided on disk with the MCM complete the following steps:
If hand entering the ladder logic by hand for the SLC, remember the following:
- Configure the slot as a 1746-BAS module in 5/02 mode
- Be sure to enter the Transfer Enable and Done bits as shown in the example logic

a) Edit the ladder logic provided on disk as needed for the application (See Section 3.0)
Verify rack and slot location in program
Modify ladder instruction addresses as needed

c) Setup the Communication Configuration parameters (See Section 4.2)
Determine each port's communication configuration requirements:
Master or Slave, Parity, Stop Bits, Baud Rate, RTS delay requirements
Identify memory mapping requirements
Set the Read Data, Write Data , and the Command Block Count parameters
Set the Slave and Master Error Table pointers are needed for the application

d) Setup the Command List if configuring a Master (See Section 4.4)
Be sure to review register map of slave device to build most effective memory map

e) Identify the module jumper requirements (See Appendix D)

f) Make up the communication cables (See Section 8). Make sure that no matter what type of connection is being made up that a jumper is in place to satisfy the CTS signal. Normally this signal will be jumpered to RTS.

g) Place processor into the run mode

h) Monitor the data table for the Master and Slave Error Status values (See Section 5.1)

'ProSoft Tested' Test Documents

Through the efforts of our 'ProSoft Tested' Program, we maintain a growing list of devices which we know have been interfaced to our module. In addition, we also have documented several of the devices which we have tested. To access this information, please visit our web site as follows:

http://www.prosoft-technology.com
Select 'Web Site Index'
Select 'MCM Connectivity Listing'
Select 'Test Document' for desired product

Table of Contents

1 Functional Overview

This section is intended to give the reader an overview of the MCM module operating concepts. Details associated with the ladder logic and the data transfer across the backplane are covered in later sections and in the Appendix.

1.1 General

The MCM products are single slot rack resident modules which have been designed to provide a tightly integrated Modbus communication interface for the Allen-Bradley 1771 and 1746 I/O platforms. The product will support the following processors:

<u>3100-MCM for 1771 Platform</u>
PLC 5 family
PLC 2 family
PLC 3 family

<u>3150-MCM for 1746 Platform</u>
SLC 5/02, 5/03, 5/04

The module will work in the local rack with the processor or can be installed in a remote rack using Remote I/O communications to link the racks, in the case of the PLC, or can be placed in an extended rack in the case of the SLC.

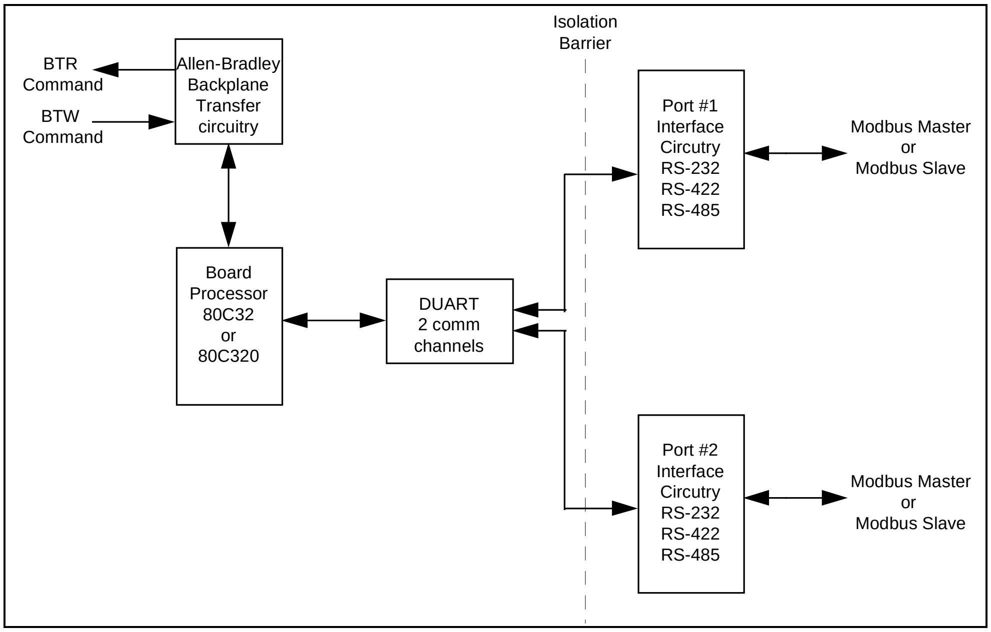
The two forms in which the product is available are shown below:

<u>3100 Module</u>
1771 Platform

<u>3150 Module</u>
1746 Platform

1.2 Hardware Overview

The design of the MCM module for the two hardware platforms are very similar. The following discussion, unless identified otherwise, will apply to both the 3100 and the 3150 platforms. The figure below shows the functional components on the modules.



Hardware layout diagram of 3100/3150 modules

The primary functional components on the boards are:

- A microcontroller responsible for the overall operation of the board, including:
  - Backplane communications with Allen-Bradley processor
  - Transferring data from module to PLC
  - Accepting data from PLC into the module
  - Servicing DUART communications
  - LED Status Indications
- An Allen-Bradley backplane chipset responsible for servicing the communications between the module and the A-B processor. The chipset contains proprietary technology licensed from A-B designed to:
  - In the case of the PLC the chipset has been designed to communicate with backplane using the Block Transfer commands, transferring 64 words at a time
  - In the case of the SLC, the chipset has been designed to communicate with the backplane using the M0/M1 files. As there is no real Block Transfer functionality in the SLC, we have implemented a form of block transfer using the I/O table to control the handshaking between the module and the processor. Up to 64 words may be transferred at a time. Shown below, presuming the module is in slot 1, are these bits:

    I:1/0 Transfer Enable
    This bit is set by the module and used by the ladder logic to enable the movement of data over the backplane

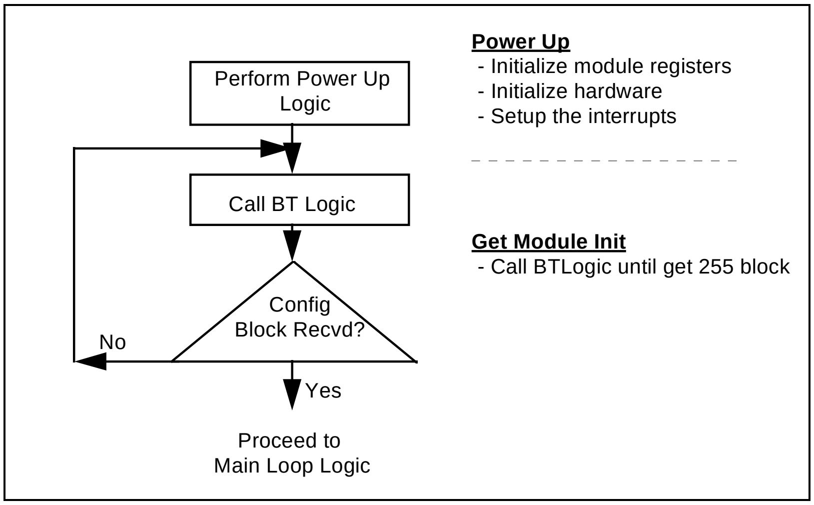
    O:1/0 Transfer Done
    This bit is set by the ladder logic to communicate to the module that the ladder has completed the data transfer
- The port interface circuitry providing the physical interface to the real world. The ports and the interface circuitry are optically isolated from the rest of the card, and therefore the backplane, providing a high level of protection to the A-B processor. Both ports are capable of supporting:
  - RS-232
  - RS-422, also called a 4 wire connection
  - RS-485, also called a 2 wire connection

1.3 General Concepts

The following discussion covers several concepts which are key to understanding the operation of the ProSoft module.

1.3.1 Module Power Up and Reset

On power up, or after pressing the reset pushbutton (3100 only), the module begins performing its logical functions. These funcions shown in the flow chart



included here, include:

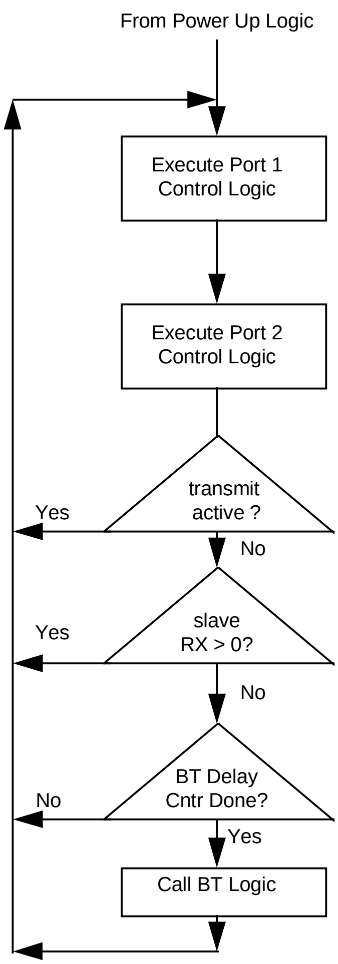
1. Initialize hardware
   - Initialize the backplane
   - Initialize the DUART
2. Initialize Module registers
   - Clear the Module Data Block
   - Clear Command List
   - Clear Error Status Tables
   - Preset constants

Once the register space has been initialized, the module will begin to block transfer with the ladder logic. The first block transfer sent from the module will initiate the configuration process, causing the ladder logic to move a 255 configuration block to the module. Once the module is configured, it will begin the Main Logic Loop.

1.3.2 Main Loop Logic

Upon completing the power up configuration process, the module jumps into an infinite loop which includes the following functions:

1. Port 1 and Port 2 handlers
   - Detect end of message condition
   - Call message handlers
   - Initiate commands if a Master
2. Block Transfer
   - Test CTS pin to assure module is not in transmit mode
   - Test slave port buffer to be sure not receiving
   - Test Block Transfer Delay counter
   - If all OK then block transfer



Port 1 Control Logic
- if port in RX mode, then test for message received
- if port in TX mode, then test if message transmit has completed
- If port is a master and ready for new command then create new command

Port 2 Control Logic
- if port in RX mode, then test for message received
- if port in TX mode, then test if message transmit has completed
- If port is a master and ready for new command then create new command

Test Transmit Status
- If either port is in process of transmitting then do not execute BT logic. The module uses the CTS pin status to detect the transmit status

Test Slave port for characters
- If either slave port is in the process of receiving characters then skip BT logic. This is done to assure timely response from the slave

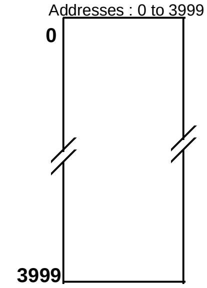
Test the Block Transfer Delay Counter
- If the Block Transfer Delay counter has incremented beyond the counter preset (set in config) then go ahead and perform block transfer

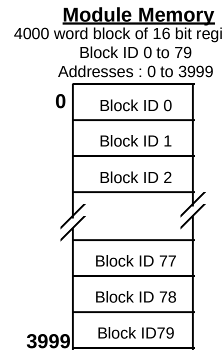
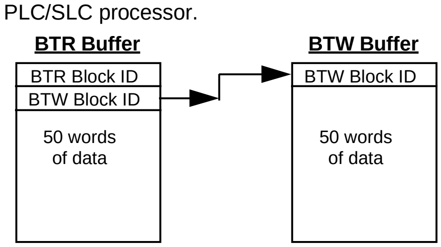
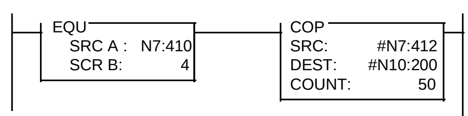
Execute Block Transfer Logic
- Calls BT Logic which executes the BTR and BTW logic

1.3.3 The Data Space in the module

One of the concepts which is important to develop an understanding of is the relationship between the data space in the module and how this data can be moved between the module and the PLC/SLC processor.

The following discussion explains the data structure in the module and how this data can be moved between the module and the ladder program. Some key points to understand:

Key Point	Description
Size of data register space in the module	The module maintains a 4000 word data space which can be used as needed by the application for data storage **Module Memory** 4000 word block of 16 bit registers Addresses : 0 to 3999 

How 4000 word module data space is broken down in blocks	This 4000 words block of data is logically broken down into 80 fifty (50) word blocks: 80 blks x 50 wrds/blk=4000 words 
How data is 'paged' between the module and processor	Via the data transfer sequence outlined in the next section, 50 word blocks of data (or 'pages') are transferred bi-directionally between the module and the PLC/SLC processor. 
How data 'page' is placed in the processor's data table	The placement of data in the PLC/SLC processor is controlled by the user and the application ladder logic. Any available data file in the processor can be used as a source of data for the module and as a destination for data from the module. 

1.3.4 The Backplane Data Transfer Process

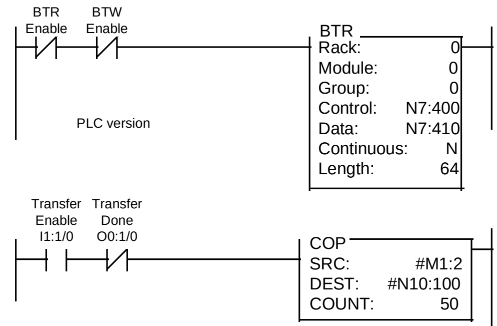
The following table provides an overview of the data transfer process between the A-B processor and the module. This process is effectively controlled by the ladder logic in the processor. The following provides some insight into the steps which occur in the module and in the ladder to effect a successful data transfer. Reference can be made to the example logic in the Appendix to see an actual implementation.

Step Number	Description

Step 1	Module generates BTR and BTW Block ID numbers based on the following logic:
	<u>BTW Block ID</u>
	if (BTW Block ID >= Write Block Cnt) then
	BTW Block ID = 80
	elseif(BTW Block ID >= 80 + Command Block Cnt) then
	BTW Block ID = Write Block Start
	else BTW block ID = BTW block ID + 1
	<u>BTR Block ID</u>
	if (BTR Block ID >= Read Block Cnt) then
	BTR Block ID = Read Block Start
	else BTR block ID = BTR block ID + 1

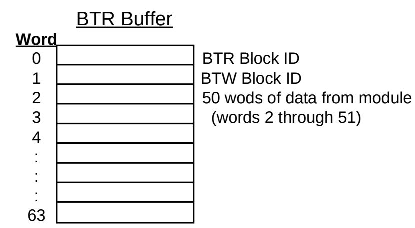
(Cont'd)

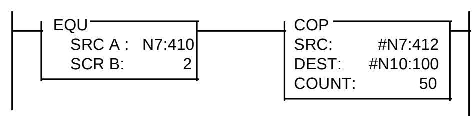
<table>
<tr><th>Step Number</th><th>Description</th></tr>
<tr><td>Step 2</td><td>Module executes a BTR command with the A-B Processor.

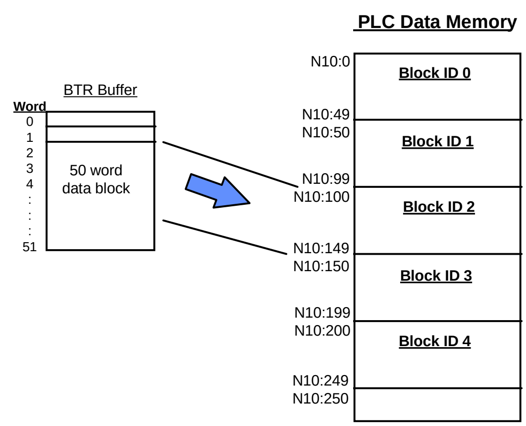


SLC version : When the Input bit goes true (the module turns this bit on), the data is ready to be copied out of the M1 file

The structure of the BTR buffer being transferred from the module is:

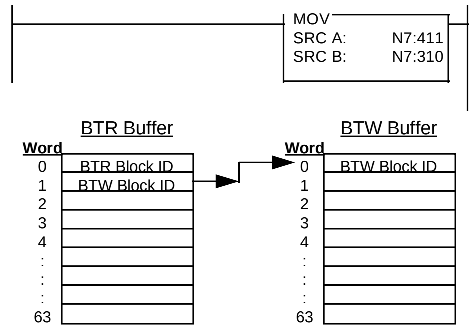

</td></tr>
<tr><td>Step 3</td><td>The ladder logic decodes the BTR Block ID and copies the data from the BTR buffer into the ladder data table based on the value of the BTR Block ID.

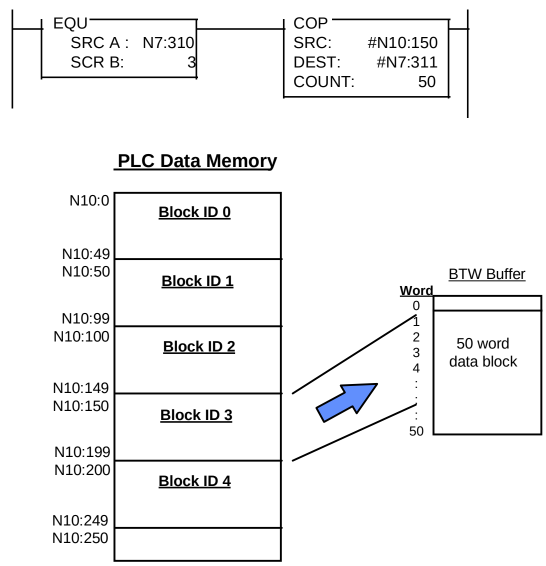


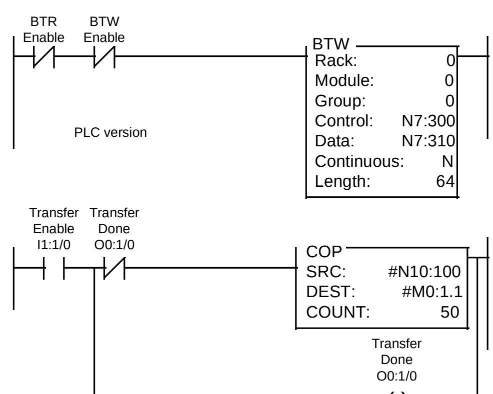

</td></tr>
</table>

(Cont'd)

<table>
<tr><td>Step 4</td><td>Transfer the BTW Block ID from the BTR Buffer to the BTW buffer.


</td></tr>
<tr><td>Step 5</td><td>Copy ladder data memory, whether Data, Command List or Configuration, to the BTW buffer. The actual data copied depends on the decoding of the BTW Block ID number.


</td></tr>
<tr><td>Step 6</td><td>Execute the BTW Command



SLC version : When the ladder logic has transferred the ladder data into the M0 file, the Transfer Done bit is set by the ladder. This bit is used by the module to determine when the transfer process is complete.</td></tr>
</table>

(Cont'd)

Step 7	The module receives the BTW data. After decoding the BTW Block ID number, the module will transfer the BTW buffer data into the correct location in the modules memory.

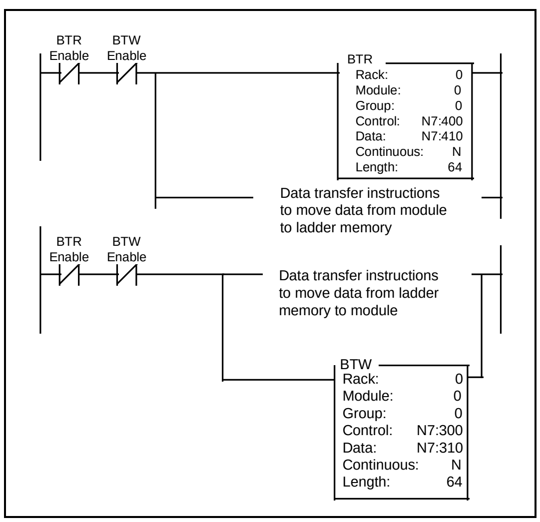
1.3.5 Interlocking the Block Transfers

One of the fundamental assumptions that the module makes is that there will be one BTR per one BTW command. In the module, upon completing the BTR instruction, the module jumps immediately to the BTW instruction. To the programmer who follows our example logic this has rather minor implications.

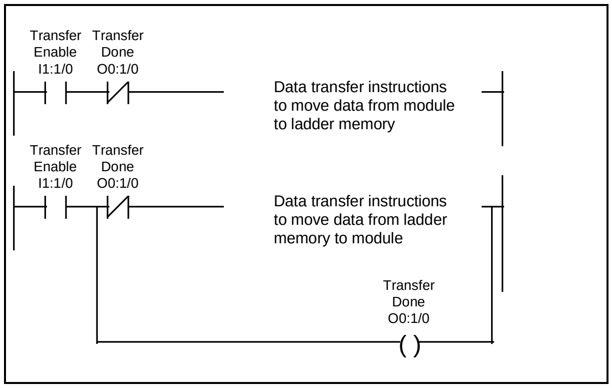
Problems arise however when a ladder logic implementation is attempted which does not meet the module's block transfer expectations. Specifically, the following must be adhered to when programming the ladder logic for the module:

<u>PLC Program using BTR/BTW Instructions</u>
In the 1771 types of processors (PLC 2, PLC 3 and PLC 5), the BTR and BTW Enable bits must be used to enable the Block Transfer Instructions. With this type of programming, the PLC is guaranteed not to execute two block transfers at the same time, and the BTR and BTW instructions are guaranteed to alternate.

Ample examples of this type of block transfer programming are available in A-B documentation as well as in the example ladder logic program in the Appendix.



SLC Program using M0/M1Instructions
In the SLC processors, there is no true mechanism for guaranteeing the integrity of data block transfers, as there is in the PLC platform. For this reason we have developed a handshaking mechanism which is designed to assure that all the words in the M0 and M1 files are transferred in unison. Following this mechanism is the only way that we can assure that the data in a block corresponds to the Block ID being transferred. The basic ladder programming which must be implemented in an SLC application is as follows:



1.3.6 SLC Processor Configuration

When initially setting up the SLC program file, or when moving the module from one slot to another, the user must configure the slot to accept the MCM module.

It is important that the slot containing the ProSoft module be configured as follows:
- 1746-BAS module with 5/02 or greater configuration
- or enter 13106 for the module ID code
- Configure the M0/M1 files for 64 words
- Configure I/O for 8 words

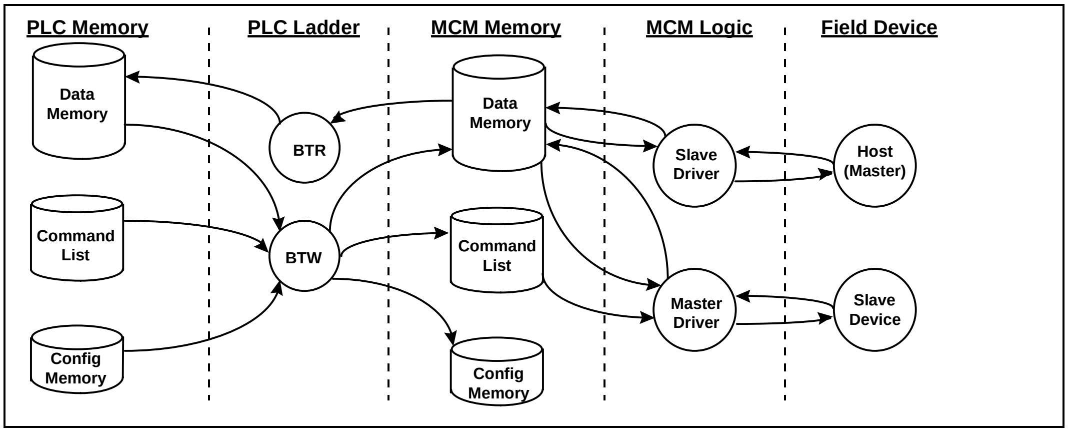
The following is a step by step on how to configure these files using Allen-Bradley APS software. Other software packages users should follow similar steps.

From the Main Menu:
1) Select the correct processor program and F3 for Off-line programming
2) F1 for Processor Functions
3) F1 for Change Processor
Modify the processor here if necessary (Note the MCM will only work with 5/02 or greater processors
4) F5 for Configure I/O
Select *1746-BAS module for SLC 5/02 or greater*, or enter 13106 for module code
5) F9 for SPIO Config when the correct slot is highlighted
6) F5 Advanced Setup
7) F5 for M0 file length - type in 64 and Enter
8) F6 for M1 file length - type in 64 and Enter
Esc out and save configuration

1.4 Data Flow

1.4.1 General concepts

In developing a solid understanding of the module's operation, it is important to understand the movement of data in between the ladder logic, the module and the Master/Slave drivers.



The following discussion covers the flow of data in the different stages. Further discussion is available in later sections on the flow of data under the different operating modes of the ports.

1.4.2 Reading data from the module

The module maintains a 4000 block of data memory. This memory contains:

1. The results of Master port transactions
2. Data moved to the module through the Slave port
3. Slave port Status data
4. Module Revision information
5. Master port Status data

During the transfer of data from the module to the PLC, the ladder logic is able to gain access to this information.

1.4.3 Writing data to the module

The module, depending on the configuration of the ports, requires three basic types of data in order to operate correctly. The three types of memory which can be transferred to the module are as follows:

1. Configuration Data. This data contains all of the parameters necessary for the module to configure the serial ports and to set up the data transfers between the module and the ladder logic
2. Command List . This set of data contains all of the parameters the module required to encode valid commands which will be transmitted out the Master port to other Modbus slave devices. Up to 20 Command List blocks can be sent to the module for a total of 100 commands
3. Data Memory. This type of memory is moved to the module to provide the data values necessary for the Slave port to service read requests (i.e., the data that a host would receive as the result of a read command), or for the Master port to service write requests (i.e., the data written to the slaves).

1.4.4 Master Port Driver

Under normal applications, the Master port is used primarily to issue read commands to slave devices, thereby acting as a data gatherer and then transferring the data which has been read to the ladder logic.

The module uses the Command List entries to encode valid Modbus commands. As each command is executed, the module scans for the next entry in the Command List. If the Master port is issuing a read command, the results of the read are deposited in the Data Memory. If the Master port is issuing a write command, data from the Data Memory is written to the slave device.

For every command which the module executes, the status of the command can be found in the Master Error Table. This table can be located anywhere in the Data Memory block and is read back into the ladder logic as part of the regular data transfer process.

1.4.5 Slave Port Driver

Operation of the Slave port can actually be a function of how the Slave port is configured. The port can operate in 3 modes:

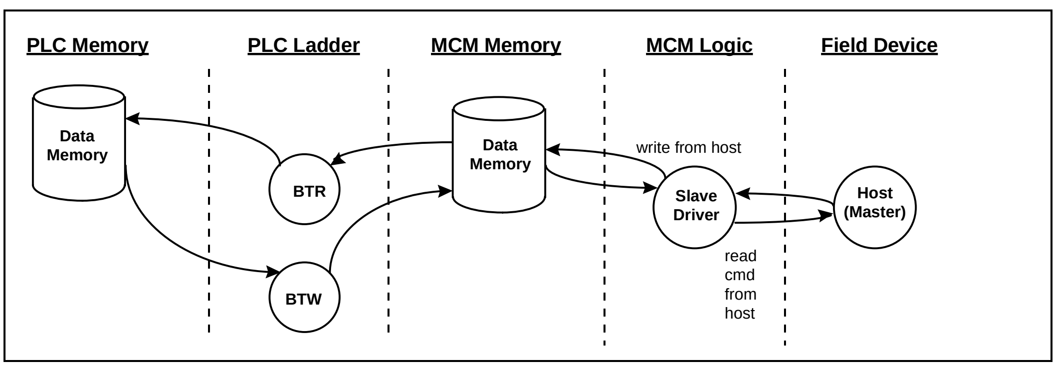
1. Normal Mode. In this mode, host read commands are serviced directly out of the Data Memory block and host write command data is sent directly into the Data Memory
2. Pass Through Mode. In this mode, hosts read request are services as in the Normal Mode, directly out of the Data Memory block. The principle difference is that host write commands are transferred to the BTR buffer for processing in the ladder logic
3. Route Mode. This mode is available when one port is configured as a Master and the other port is a Slave . In this mode, the Slave driver checks every command's slave address against up to six entries in the Routing Table (user setup). If there is a match, the command which has been received on the slave port is routed out the master port. The response from the slave is received by the module and transferred to the slave port for transmission to the host.

The following sections provide some data flow diagrams to assist in understanding the different modes.

1.4.6 Slave Port - Normal Mode

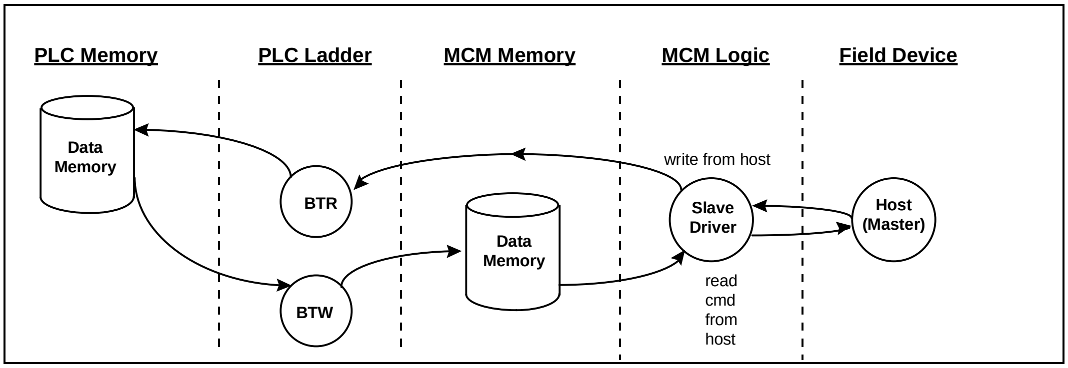
Under normal operation (i.e., no special features enabled) the Slave port services all of received read and write requests using the Data Memory. This offers the advantage that it is very simple to setup and requires very little ladder logic to implement.

The primary requirement is that the data space which the host is writing into must not overlap with the data space which the ladder logic is writing into. Under many conditions, this limitation does not pose a problem, in which case this is the best mode in which to configure the module.



1.4.7 Slave Port - Pass Through Mode

The Pass Through Mode offers the advantage that it overcome the one limitation imposed by the Normal Mode of operation. As mentioned above, if host write commands received by the slave port must overlap with the address space which the ladder logic is also writing into, then the Normal Mode cannot be used.



In the Pass Through mode, all write commands received by the module (addressed to the local slave address or broadcast) are transferred into the BTR buffer for handling by the ladder logic. This mode offers several advantages:

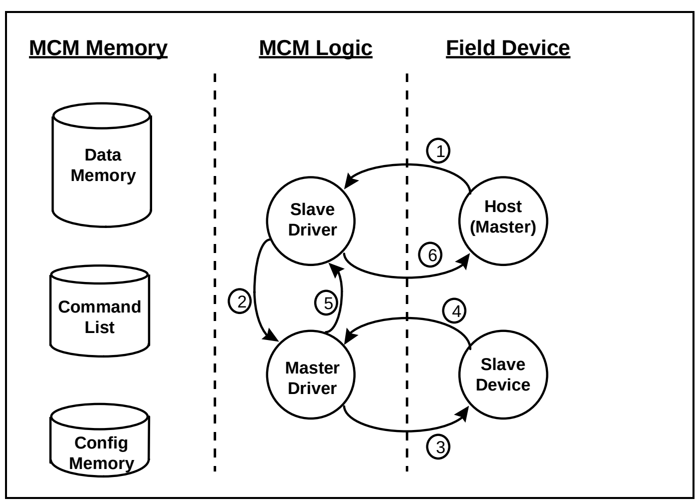
1. The ladder data table can be accessed directly by the host commands
2. All write commands can be conditionally accepted by implementing ladder logic which limits such things as address range, data value limits, etc.
3. The ladder program will know definitively when the slave port has received a write command from the host. In the Normal Mode, the ladder logic cannot differentiate between the types of commands which have been handled.

The only consideration to implementing the Pass Through Mode is that the ladder logic requirements are a bit more substantial than those of the Normal Mode.

1.4.8 Slave Port - Route Mode

The Route Mode can be a powerful tool if a host must have periodic access to data from a slave connected to the module' master port. This can be a common requirement in SCADA types of applications where there may be some other Modbus devices connected to the Master port (such as Flow Computers) which have large volumes of data useful to a host but not required by the local PLC/SLC.

When the Slave driver detects that a command has been received for a device in the module's Route List (configured during module setup), the following steps occur:



1. The host issues a read or write command to a slave address matching one of the entries in the MCM module's Route List
2. The MCM Slave driver detects the match with the entry in the Route List and sends the command to the Master driver

3. The Master driver executes the command at the soonest possible time (i.e., as soon as any currently executing command transaction is completed)
4. The addressed slave responds to the command
5. The slave's response is passed from the Master driver to the Slave driver
6. The Slave driver returns the response to the Host

1.5 Modbus Addressing

When applying the MCM module, whether as a Master or as a Slave, it is important to understand, to at least a minimum level, the issues associated with Modbus addressing. This section provides a primer on Modbus addressing to familiarize new Modbus users with terminology and concepts. If more is desired on the subject of Modbus, excerpts from the Modbus Protocol Specifications are available in the Appendix.

1.5.1 Modbus Addressing Concepts

The Modbus addressing scheme was developed early on around the data table and I/O structure in Modicon PLCs. As a result, the Modbus protocol supports access to the various data spaces in the Modicon PLC.

By far the most common data space used is the 4xxxx space using the Function Codes 3, 6 and 16. This space is used to transfer 16 bit register values, floating point values, and even bit mapped data. Using formal Modbus addressing terminology, this data space actually starts at address 40001 (Modbus is one (1) based, while the MCM data table addressing is zero (0) based).

Access to the different data spaces is determined by the Function Code which is used. The following chart shows the four different types of data spaces, the numerical range of these spaces, and the function codes which are used to execute read and write instructions within these data spaces.

Register Space			Read Command	Write Command
0XXXX	Coils (Discrete Out)	1 9999	1	5,15
1XXXX	Inputs (Discrete In)	1 9999	2	N/A
3XXXX	Input Registers (Analog In)	1 9999	4	N/A
4XXXX	Holding Registers (Memory Regs)	1	3	6,16

Relationship between the Modbus Function Codes and the Modbus addressing scheme

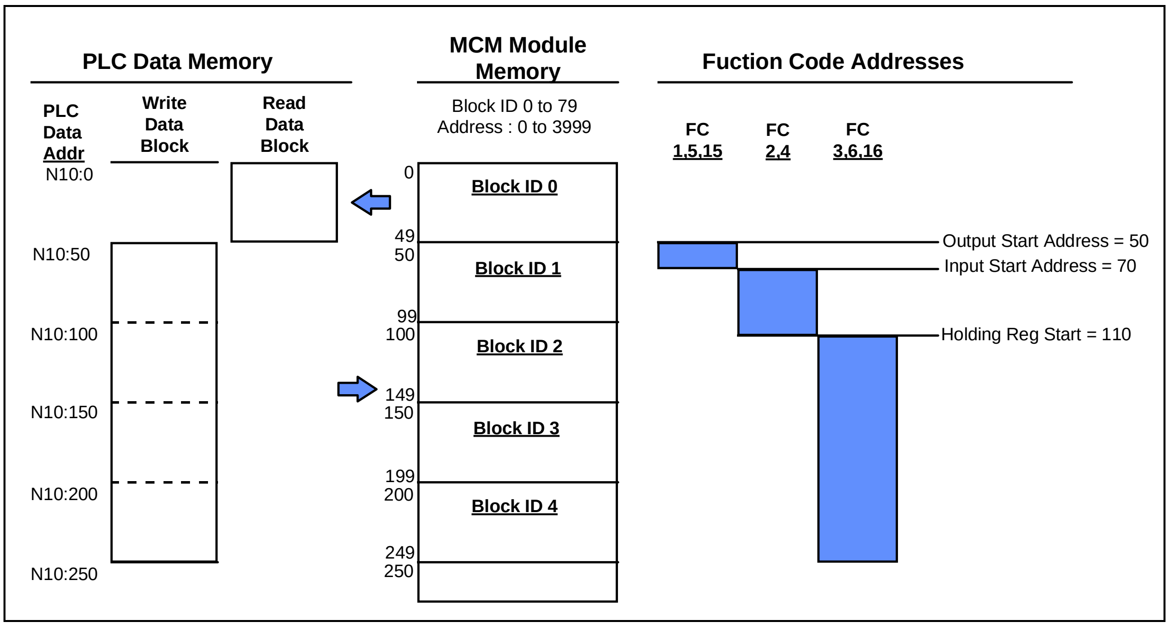
1.5.2 MCM Support of Modbus Functionality

The MCM module supports all of the common Modbus Function Codes used for data transfer. The following table documents the Function Codes and the addressing ranges which are supported (See Section 6 for more details).

Function Code	Description	Address Range
1	Read Output Status	0001 - 9999
2	Read Input Status	10001 - 29999
3	Read Holding Regs	40001 - 49999
4	Read Input Regs	30001 to 39999
5	Force Single Coil	00001 - 9999
6	Preset Single Reg	40001 - 49999
8	Loopback Test	
15	Force Multiple Coils	0001 - 9999
16	Preset Multiple Regs	40001 - 49999

1.5.3 Mapping Modbus Addresses to Ladder Data Addresses

The relationship between the ladder logic and the Modbus addressing scheme is controlled by the Application Programmer. The following diagram is an example of the memory map which could be setup in an application involving the module.



In this example, the MCM is setup as a Modbus Slave responding to queries from a Host. The Host will read data from the module starting at module register 50. The following memory table will be true for this layout:

Function Code	Module Address	Modbus Address
1,5,15	50 to 69	1 to 320
2, 4	70 to 109	10001 to 10640 30001 to 30040
3, 6, 16	110 to 249	40001 to 40140

2 Getting Going - A Step by Step Approach

Installation of the 3100/3150-MCM module is easily accomplished. Installation into a system requires only a few steps. Following is a step-by-step procedure for getting an application operational:

Step	Example	User Application
1. Identify Rack position	Rack 0 Group 2 Slot 0	Rack : ____ Group : _____ Slot : ______
2. Identify PLC Data Files usage	BT Buffers: N7 BT Control: N7 Config File: N7 Data File : N10	BT Buffers: N____ BT Control: N____ Config File: N____ Data File : N_____
3. Ladder Logic	Example on disk and in Appendix (Several examples to choose from)	Select the example closest to your application and modify as needed
4. Modify Logic for rack position	PLC BTR - Rung 2:0 BTW - Rung 2:1 SLC I:x.0 addresses O:x.0 addresses M0:x addresses M1:x addresses	Modify these instructions as needed based on the required rack position. Be sure to configure the slot in the SLC
5. Modify Logic for Data file usage	N7 and N10 is used as data space for the module	Create files and change references from N7 and N10
6. Install card in rack	Power down rack and install module	Power down and install module
7. Connect a comm cable to the front of the module	Decide on cable type needed for application	
9. Apply power to system and place PLC in RUN	Monitor the status files and the LEDs on the front of the module	

Once the hardware has been installed and the necessary programming has been downloaded to the processor, the system is ready (Presuming all other system components are safely ready).

3 Ladder Logic Overview

Data transfers between the processor and the ProSoft Technology module occur using the Block Transfer commands, in the case of the PLC, and M0/M1 data transfer commands, in the case of the SLC. These commands transfer up to 64 physical registers per transfer. The logical data length changes depending on the data transfer function.

The following discussions and Sections details the data structures used to transfer the different types of data between the ProSoft Technology module and the processor. The term 'Block Transfer' is used generically in the following discussion to depict the transfer of data blocks between the processor and the ProSoft Technology module. Although a true Block Transfer function does not exist in the SLC, we have implemented a pseudo-block transfer command in order to assure data integrity at the block level. Examples of the PLC and SLC ladder logic are included in the Appendix.

In order for the ProSoft Technology module to function, the PLC/SLC must be in the RUN mode, or in the REM RUN mode. If in any other mode (Fault/PGM), the module will stop all communications until block transfers resume.

3.1 Operational Overview

On power up the module moves a 255 into Word 1 of the BTR data file. This is a signal that the module needs to receive configuration data before proceeding any further. Once the configuration is received the module will begin transferring data to and from the processor depending upon how many Read and Write block counts have been configured. Once these are completed, the module will then transfer the command blocks if any have been configured.

3.2 Ladder Logic

The flow of the ladder logic is somewhat predefined by the way the module has been programmed. The expected flow of the ladder logic should be as follows:

<u>Read Rung</u>

1. Read Data from the Module. In the case of the PLC the module data will be transferred into the BTR Buffer. In the case of the SLC the module data will be accessed directly out of the M1 file
2. Decode the BTR Block ID number. Depending on the value of the BTR Block ID, copy the module data into the correct location in the ladder logic data table
3. Move the BTW Block ID Number from Word 1 of the BTR Buffer into Word 0 of the BTW Buffer. In the case of the SLC the transfer will actually be from Word 1 of the M1 file to Word 0 of the M0 file. The BTW Block ID number should be manipulated if necessary to assure that data is not overwritten in the module (The LIM test branch does this in the example logic)
4. Test for Event Initiated Commands and module configuration

<u>Write Rung</u>

1. Decode the BTW Block ID number and depending on the value move either data values, Command List values or Configuration values to the BTW buffer (M0 file in the SLC)
2. If the configuration transfer is enabled, then clear the configuration enable bit
3. In an Event Initiated Command is enabled, then clear the enable bit
4. Execute the BTW transfer. In the PLC this will be done by enabling the BTW instruction. In the SLC, this will be done by setting the Transfer Done bit (an Output bit has been assigned to this function in the design of the module)

4 Writing to the Module

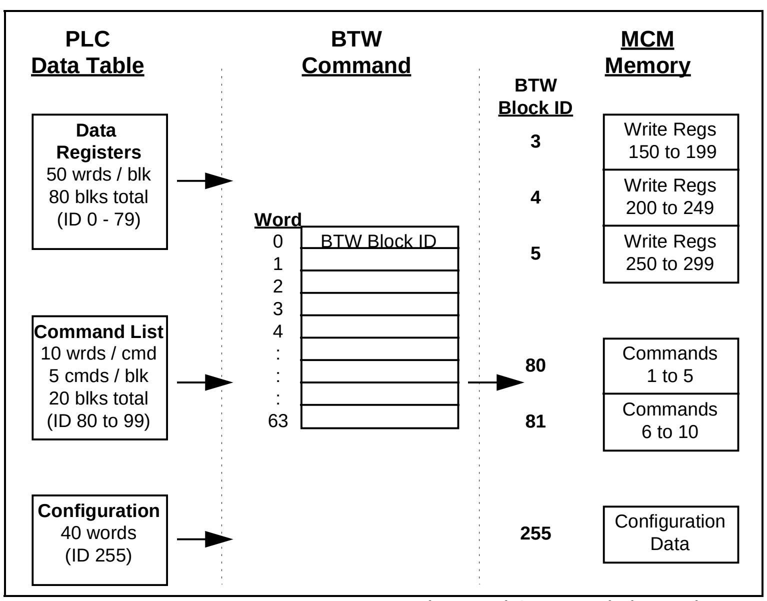
This section provides reference level details on the transfer of data from the PLC/SLC processor to the MCM module. This type of transfer allows the ladder logic to send configuration , command list and data to the module.

4.1 Block Transferring to the Module

Data transfer to the module from the processor is executed through the Block Transfer Write function. The different types of data which are transferred require slightly different data block structures, but the basic data structure is:

Word	Name	Description
0	BTW Block ID	A block page identifier code. This code is used by the ProSoft module to determine what to do with the data block. Valid codes are: BTW Code — Description 0-79 — Module Data Memory 80-99 — Command List 100-120 — Event Driven Writes 255 — Module Communication Configuration
1 to 63	Data	The data to be written to the module. The structure of the data is dependent on the Block ID code. The following sections provide details on the different structures.

Although the full physical 64 words of the data buffer may not be used, the BTW and M0 lengths must be configured for 64 words, otherwise module operation will be unpredictable.



***Data transfer from PLC to MCM** : Data values and Command List entries are 'paged' into the MCM module. The data type and location being written into corresponds to the BTW Block ID number. The BTW Block ID number is controlled by the MCM module, as discussed later in this section.*

4.2 Communications Configuration [BTW Block ID 255]

The ProSoft Technology firmware communication parameters must be configured at least once when the card is first powered up, and any time thereafter when the parameters must be changed.

Power Up

On power up, the module enters into a logical loop waiting to receive configuration data from the processor. While waiting, the module sets the second word of the BTR buffer (the BTW Block ID) to 255, telling the processor that the module must be configured before anything else will be done. The module will continuously perform block transfers until the communications configuration parameters block is received. Upon receipt, the module will begin execution of the command list if present, or begin looking for the command list from the processor.

Changing parameters during operation

Changing values in the configuration table can be done at any time. The module does not accept any of the changes until the 're-configuration' process is initiated. This can be accomplished in several ways, including:

1. Cycle power to the rack
2. Press the reset pushbutton on the module (3100 only)
3. Move 255 into BTW Block ID position (See example logic when B3/0 is set)

During this process, the 'CFG' LED will toggle, giving a visual indication that the module has received the configuration block.

Transferring the Communications Configuration Parameters to the module will force a reset of the communication port, as well as dropping DTR for 200 ms pulses to reset any attached hardware.

The configuration data block structure which must be transferred from the processor to the module is as follows:

BTW Buffer	Data Addr	Name	Example Value
0		BTW Block ID	255
		Port 1 Config	
1	N[ ]:0	Port Configuration Word	0 - Master 1 - Slave
2	N[ ]:1	Port Slave Addr	1
3	N[ ]:2	Baud Rate	5
4	N[ ]:3	RTS to TxD Delay	0
5	N[ ]:4	RTS Off Delay	0
6	N[ ]:5	Response Timeout	0
7	N[ ]:6	Intercharacter Delay	0
8	N[ ]:7	Setup Parm #1	0
9	N[ ]:8	Setup Parm #2	0
10	N[ ]:9	Setup Parm #3	0
		Port 2 Config	
11	N[ ]:10	Port Configuration Word	0 - Master 1 - Slave
12	N[ ]:11	Port Slave Addr	1
13	N[ ]:12	Baud Rate	5
14	N[ ]:13	RTS to TxD Delay	0
15	N[ ]:14	RTS Off Delay	0
16	N[ ]:15	Response Timeout	0
17	N[ ]:16	Intercharacter Delay	0
18	N[ ]:17	Setup Parm #1	0
19	N[ ]:18	Setup Parm #2	0
20	N[ ]:19	Setup Parm #3	0

(Cont'd)

		System Configuration	
21	N[ ]:20	Read Block Cnt	3
22	N[ ]:21	Write Block Cnt	1
23	N[ ]:22	Cmd Block Cnt	2
24	N[ ]:23	Slave Err Ptr	100
25	N[ ]:24	Master Error Ptr	120
26	N[ ]:25	BT Delay Cntr	0
27	N[ ]:26	Floating Point Offset	0
28	N[ ]:27	Read Block ID Start	0
29	N[ ]:28	Write Block ID Start	0
30	N[ ]:29	Spare	0
31 - 36	N[ ]:30 to N[ ]:35	Route Mode Slaves 1 to 6	0

Port 1 and 2 Configuration

Data Addr	Name	Description
N[ ]:0 N[ ]:10	Port Configuration Word	This register contains several communication configuration parameters encoded into the word. These are as follows: Protocol Mode: The port's protocol mode is selected by these bits: **Bits** **210** 000 Modbus Master - RTU Mode 001 Modbus Slave - RTU Mode 010 Modbus Master - ASCII Mode 7 bit 011 Modbus Slave - ASCII Mode 7 bit 100 Modbus Master - ASCII Mode 8 bit 101 Modbus Slave - ASCII Mode 8 bit Pass Through Mode: The Slave Port operating mode is selected by this bit: **Bit** **3** 0 Pass Through Disabled 1 Pass Through Enabled Routing Mode: Enable the Slave to Master Routing mode: **Bit** **4** 0 Routing Mode Disabled 1 Routing Mode Enabled Stop Bits: The number of stop bits to be used is defined as follows: **Bits** **13 12** 0 0 One stop bit 0 1 Two stop bits 1 x Invalid Port Configuration Parity: The parity mode to be used by the module is defined by this word as follows: **Bits** **15 14** 0 0 No parity 0 1 Odd parity 1 0 Even parity 1 1 Invalid Port Configuration

(Cont'd)

N[ ]:1 N[ ]:11	Slave Address	When the port is configured to operate in the Slave mode, the value entered in this register is used as the Modbus Slave address. Valid values range from 1 to 247.
N[ ]:2 N[ ]:12	Baud Rate	The baud rate at which the port is to operate. The available configurations are as follows: **Value** / **Baud Rate** 0 — 300 Baud 1 — 600 Baud 2 — 1200 Baud 3 — 2400 Baud 4 — 4800 Baud 5 — 9600 Baud 6 — 19200 Baud 7 — 38400 Baud The module's two ports are limited to an upper baud rate of either 19200 or 38400 baud. The module cannot be configured with one port at 19200 and the other at 38400. If an attempt is made to configure the module in this fashion, a Port Configuration Error will be returned.
N[ ]:3 N[ ]:13	RTS to TXD Delay	This value represents the time in 1 ms increments to be inserted between asserting RTS, and the actual transmission of data. The delay, if greater in duration than the hardware time delay associated with CTS, will override the CTS line until the time-out is complete. This configurable parameter is useful when interfacing with modem based devices, anytime line noise must be allowed to subside before data is transmitted, or if data transmittions must be slowed down. Valid values range from 0 to 65535 (0xffff).
N[ ]:4 N[ ]:14	RTS Off Delay	The value in this word represents the number of 1 ms time delay increments inserted after the last character is transmitted and before RTS is dropped. The module automatically inserts a one character width Off Delay, assuring that RTS does not drop until after the last character has been completely sent. Unless working under unusual conditions, this value will normally be configured with a value of 0. Valid value range from 0 to 65535 (0xffff).
N[ ]:5 N[ ]:15	Message Response Timeout	This register represents the message response timeout period in 1 ms increments. This is the time which a port configured as a Master will wait before re-transmitting a command if no response is received from the addressed slave. The value is set depending on the expected slave response times. The allowable range of values is 0 to 65535(0xffff). If a zero value is entered, the module will default to a one second timeout value (1000 ms).
N[ ]:6 N[ ]:16	Inter-character Timeout	This register is used in situations where the end of message character timeout delay must be extended beyond the normal 3.5 character widths. The value entered represents the number of 1 ms intervals of 'no transmission' which will be counted prior to accepting a message. This parameter will be useful in satallite or packet radio installation where a data transmission may be split between two packets. Increasing this value beyond the system's packet handling time will eliminate timeout errors. Valid values range from 0 to 65535 (0xffff)

(Cont'd)

N[ ]:7 N[ ]:17	Setup Parameter #1 - Master Mode : Not Used - Slave Mode : Input Memory Start Address	Modbus Slave Mode: This value defines the offset address into the 4000 word data space that the MCM Slave port will use when responding to function code 2 and 4 commands. As an example, to start the address space at word 150, enter a 150. A function 2 or 4 command with an address of zero (10001 or 30001) will then start reading at word 150. Valid values range from 0 to 3999.
N[ ]:8 N[ ]:18	Setup Parameter #2 - Master Mode : Not Used - Slave Mode : Output Memory Start Address	Modbus Slave Mode This value defines the offset address into the 4000 word data space that the MCM Slave port will use when responding to the function code 1, 5 or 15 commands. As an example, to locate the output image at word 100, enter a 100. A function code 1 command with an address of zero (1) will then start reading at word 100. Valid values range from 0 to 3999.
N[ ]:9 N[ ]:19	Setup Parameter #3 - Master Mode : Not Used - Slave Mode : Holding Register Start Address	Modbus Slave Mode This value defines the offset address into the 4000 word data space that the MCM Slave port will use when responding to the function code 3, 6, or 16 commands. As an example, to locate address 40001 at word 100 in the module, enter a 100. A function code 3 command with an address of zero (40001) will then start reading at word 100. Valid values range from 0 to 3999.

System Configuration

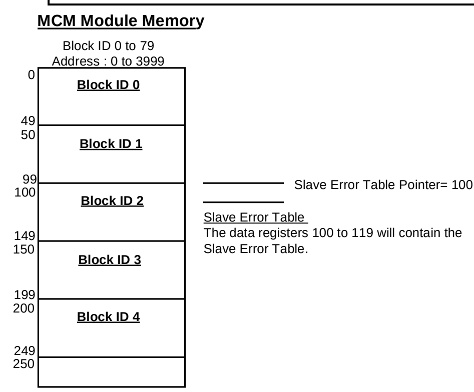
Data Addr	Name	Description
N[ ]:20	Read Data Block Count	This value represents the number of 50 word data blocks which are to be transferred from the MCM Module to the processor. The blocks returned from the module start at block 0 and increment from there. The maximum block count is 80. As an example, a value of 5 will return BTR Block ID data blocks 0, 1, 2, 3, and 4, or module registers 0 to 249. If a value greater than 80 is entered, a System Configuration Error is activated
N[ ]:21	Write Data Block Count	This value represents the number of 50 word data blocks which are to be transferred from the processor to the MCM Module. The module will use this value to return a BTW Block ID Number to the processor. The ladder logic can use this value to determine which data to move to the MCM via the Block Transfer Write. The maximum block count is 80. As an example, if a value of 5 is entered, the MCM will return BTW Block ID numbers 0, 1, 2, 3, and 4 to the ladder logic (See Section 4.2). If a value greater than 80 is entered, a System Configuration Error is activated

(Cont'd)

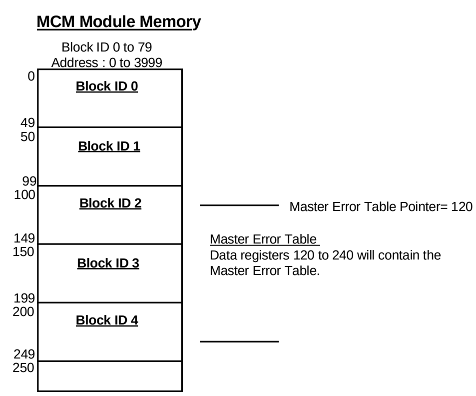
<table>
<tr><td>N[ ]:22</td><td>Command Block Count</td><td>This value represents the number of 50 word Command Blocks which are to be transferred from the processor to the MCM Module. This value will be 0 if the module will not be configured with a Master port. See the discussion in Section 4.1.3 for details on the number of Command Blocks needed. The maximum block count is 20.

If a value greater than 20 is entered, a System Configuration Error is activated</td></tr>
<tr><td>N[ ]:23</td><td>Slave Error Block Pointer</td><td>This value represents the relative starting position in the module's data table within which the Modbus Slave Error Data Block is placed. The Slave Error Table is a 20 word block containing Slave port status and several communication counters. The error data can be placed anywhere in the module's data space (0 to 3999). The contents of the Error Table can then be obtained as part of the regular Register Data.

If a value greater than 3980 is entered, a System Configuration Error is activated


</td></tr>
<tr><td>N[ ]:24</td><td>Master Error Block Pointer</td><td>This value represents the relative starting position in the module's data register table within which the Master Error Data Block is placed. The error block (120 words in length) can be placed anywhere in the module's data space (0 to 3999). The contents of the Error Table can then be obtained as part of the regular Register Data.

If a value greater than 3880 is entered, a System Configuration Error is activated


</td></tr>
</table>

(Cont'd)

N[ ]:25	Block Transfer Delay Counter	This is an empirical value used by the module to balance the amount of time the module spends block transferring and the amount spent handling port communications. The value entered is used as a loop counter in the module, where each time through the loop the count is incremented. When the count equals the Block Transfer Delay Counter a Block Transfer sequence is initiated. The range on this value is 0 to 255. <u>Example</u> : In Master Mode applications with the module in a remote rack, the frequency of command execution can be improved by entering a value of 75-150. The value must be determined empirically.
N[ ]:26	Floating Point Offset	This value is used by the module's Slave port driver to support the read and write addressing of Floating Point registers when addressing registers > 7000 (Commonly called the Enron version of the Modbus protocol). The offset value is used as follows by the module: MCM Reg Address = Floating Point Offset + (Reg Addr - 7000) * 2 The Floating Point Offset is not used on Pass Through Mode addressing. In the Pass Through mode, the address passed to the ladder logic is calculated as follows: Address = Reg Addr - 7000
N[ ]:27	Read Block ID Start	This value determines the starting BTR Block ID number which will be returned from the module. As an example, if the ladder logic needs to receive Blocks 2 through 5 from the module, the parameter should be configured with a '2' and the Read Block Count should be set to '4'. Valid values range from 0 to 79.
N[ ]:28	Write Block ID Start	This value determines the starting BTW Block ID number which the module will return to the ladder logic. As an example, if the ladder logic needs to write into Blocks 4 through 5 in the module, this parameter should be set to '4' and the Write Block Count should be set to '2'. Valid values range from 0 to 79.
N[ ]:30 to N[ ]:35	Route Mode Slave Address #1-#6	These six addresses are provided for when the MCM module is configured with the Routing Mode Enabled. In this mode, any command which comes in the Slave port which matches one of the Route Mode Addresses will be re-transmitted out the Master port. The response from this slave will be routed back to the host via the slave port.

4.3 Writing Into Module Data Memory [BTW Block ID Codes 0-79]

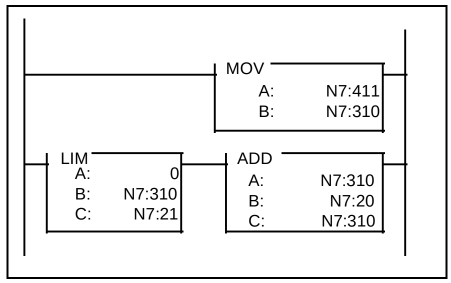
Writing into the MCM register data space is accomplished using a Block Transfer Write with BTW Block ID codes from 0 to 79 followed by 50 words of data.

Care must be exercised with memory layout to assure that MCM read and write commands do not overwrite data being moved in from the processor ladder logic. Modbus data <u>**cannot**</u> be moved into a 50 word block that is also updated by the processor. The ladder logic examples in the Appendix address this concern.

4.3.1 Ladder Logic to Write Data to Module

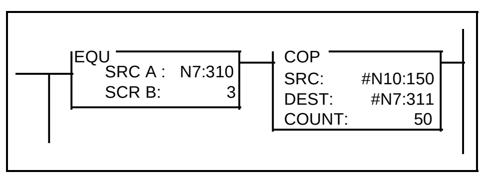
The ladder logic required to move data to the module is a simple series of EQU-COP branches, or it can be implemented using indirect addressing. The way that we have implemented the transfer to the module in all of our example ladder logic (See Appendix and Application Notes) is through a two step process, where:

Step 1 : During the BTR process, the module will 'feed' the ladder logic a BTW Block ID Number in the second word of the BTR Data Buffer. Ladder logic is implemented to accept this value, condition it if needed, and then to move the value to the actual BTW Block ID location. The ladder logic to do this is shown below:



Setting up the BTW Block ID Number
Located at the bottom of the BTR rung (Rung 0), this logic moves the BTW Block ID Number being received from the module and offsets it by the Read Block Count (N7:20) in order to assure that PLC data does not overwrite the data being returned from the module to the PLC. See logic in Appendix for implementation details.

Step 2 : During the processing of the BTW rung, the ladder logic will test for the value in the BTW Block ID register and based on the value, copy data from the data table into the BTW Block Transfer buffer. This process requires that every BTW Block ID which will be processes be accounted for with a branch of logic. An example of the ladder logic required follows:



Test BTW Block ID and move data to BTW Buffer
This branch, located in the BTW rung (rung 1) is an example of the logic that must be implemented for each data block to be move to the module. See logic in Appendix for implementation example.

4.3.2 Block Transfer Data Structure

The structure of the block transfer buffer when writing data to the module is shown below:

Word	Name	Description
0	BTW Block ID	The block identifier number allows the MCM Module to decode which '50 word page' in the module's 4000 word data space the data is to be written. The data space to be written into can be determined by multiplying the BTW Block ID by 50. The result is the first word of the 'page'. As an example: **BTW** **Block ID** **Data Space** 0 0 to 49 1 50 to 99 10 500 to 549 20 1000 to 1049 By paging the different data blocks into the module the processor can control the module data memory contents.
1 to 50	Data	The data to be written to the module.

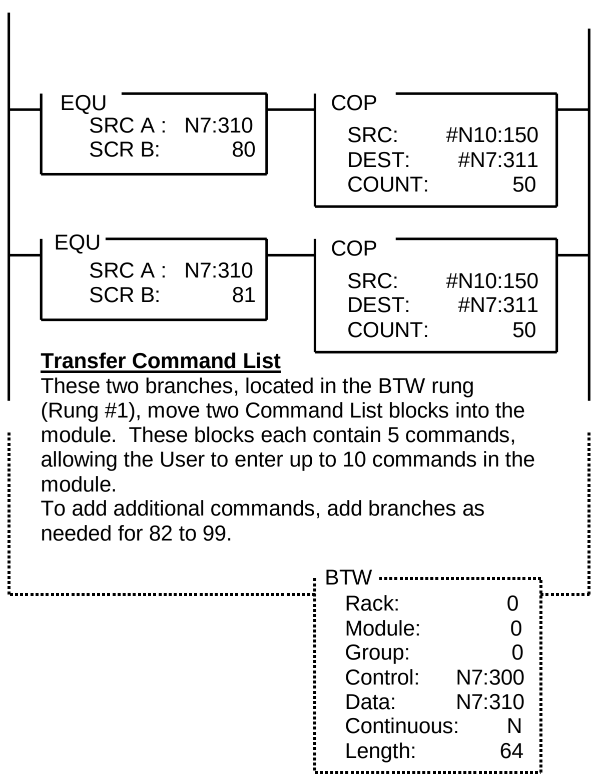
4.4 Command List Configuration - Master Mode [BTW Block ID Codes 80-99]

An MCM Modbus Master port establishes communications and performs various communications functions based on the data which the user has placed in the command list. The command list consists of up to 100 individually configured command data blocks (10 words reserved per command) which are shared between the two available ports (in the case when the module is configured with two Master ports).

4.4.1 Command List Ladder Logic

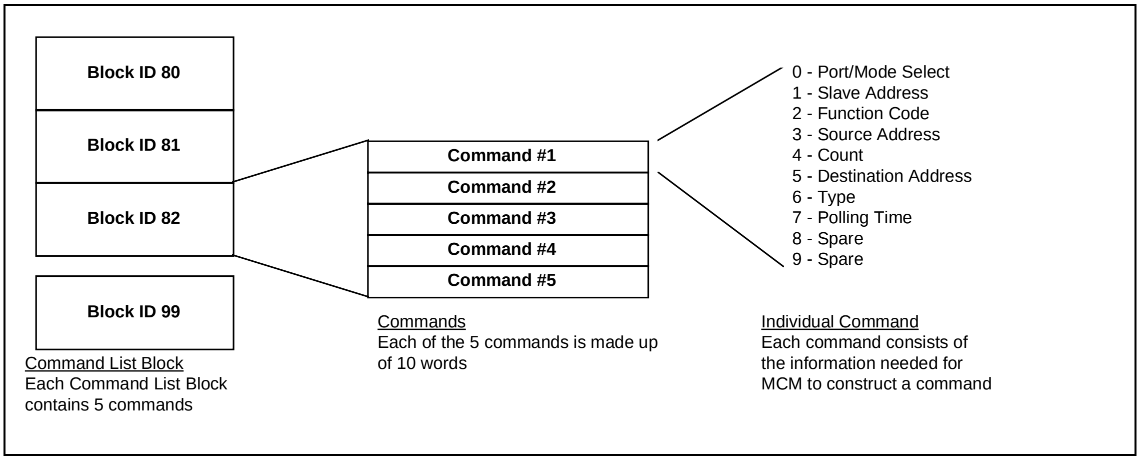
This list, entered into the processor Data Table, is transferred to the module's memory using BTW Block ID codes 80-99 with each code representing a 50 word block, or 5 commands.

An example of the ladder logic to move the commands to the module is as follows:



4.4.2 Command List Structure

The structure of the block containing the Command List is shown in the diagram below:



See Section 6 for details on configuring Modbus Commands

Name	Description
Port/Mode Select	The Port/Mode Select parameter allows the application to select which port the MCM Module will use to execute the command, and whether the command will be performed continuously or only when a change in data is detected (Conditional), or under direct ladder logic control (Control). Valid values are: **Port/Mode** — **Description** 0 — Disable Command 1 — Port 1 Continuous Command 2 — Port 2 Continuous Command 5 — Port 1 Conditional Command 6 — Port 2 Conditional Command 9 — Port 1 Control Command 10 — Port 2 Control Command Continuous vs. Conditional : Function Codes 5, 6, 15, and 16 When configuring write commands in the Command List, the MCM Master driver can support two types of data write commands; Continuous and Conditional. The difference between the two are: Continuous : Commands entered as in this fashion will be executed every time the module's Command List is scanned. Conditional: Conditional commands are executed only when a change in the block of data to be written is detected. Every time the Command List is scanned, the module will compare the data to be written against the data last written. If a change is detected in any value, bit or word, the entire data block controlled by the command is written. Control Command Mode In the Control Command Mode, the command will only be executed when the Command Enable Bit (see Section 4.5) transitions from 0 to 1. The command is executed once per transition (i.e., the module performs some one-shot logic to assure that the command only executes one). To clear the one-shot in the module, the Command Enable Bit must change state from 1 back to 0.

(Cont'd)

Slave Address	The slave address represents the Modbus slave address of the slave station to which the command is directed. Addresses should be entered in the decimal form.
Function Code	The function code entered in the table tells the MCM Module what command to execute. The different choices are detailed in Section 6, but in an overview they are as follows: **Function** **<u>Code</u>** **<u>Description</u>** 1 Read Output Status 2 Read Input Status 3 Read Multiple Data Registers 4 Read Input Registers 5 Force Single Coil (Latch/Unlatch) 6 Preset (Write) Single Data Register 15 Multiple Coil Latch/Unlatch 16 Preset (Write) Multiple Data Register
Source Address	The value represents the register or bit address, for both read and write commands, from which data will be obtained. The distinction between the two is as follows: - When issuing a read command, the Source Register Address is the register location in the slave where the command will begin getting data - When issuing a write command, the Source Register Address is register in the module where the command will begin obtaining the data to be written to the slave.
Count	The number of words or bits the Modbus command is to read or write. See Section 6 for a detailed discussion on the word and bit lengths to be specified for the different commands.
Destination Address	The value represents the register or bit address, for both read and write commands, to which data will be written. The distinction between the two is as follows: - When issuing a read command, the Destination Address is the register location in the module where the command will begin placing the data from the slave - When issuing a write command, the Destination Address is register in the slave where the command will begin placing the data to be written to the slave.
Type	The Type field is relevant only during a Function Code 3 command (Multiple Register Read). The Type field tells the module to execute word swapping on the data being received by that particular command. This is extremely useful and important when reading floating point data (two words per value) from some instruments (Some instruments store the words of their floating point data in the opposite orientation to that of the processor. In these case, swapping the words allows a ladder logic COP command to copy the data straight from an Integer file to a Floating Point file). The available options at this time are: **<u>Type</u>** **<u>Description</u>** 0 Default value. Performs no swapping. 1 Swap words in each word pair received from the slave during this command 2 Swap words in each word pair received then swap the bytes within the words 3 Swap bytes within each word (no word swapping) If using a function 6 or 16 and the destination register is greater than 47000, use a 1 in this field to disable the Enron Extension.

Polling Time Preset	The Polling Time Preset value allows each command to have a configurable execution frequency. In the module, a timer is maintained for each command. Once per second the timer is decremented, until it reaches zero. When the timer reaches zero, the command is enabled for execution, and the timer is reset to the Polling Timer Preset value. The resolution of the polling timer is 1 second. Valid values are 0 to 65535 (0xffff).

4.4.3 Editing the Command List

Entering the Command List is a matter of entering the correct values into the PLC data table. Using the ladder logic programming software, enter the values necessary to setup one or more valid commands.

Hints to Make Life Easier

When first setting up the Command List we recommend that you start out with one command. This one command will allow the module to begin transmitting if all else is OK (i.e., ladder logic, cable is connected, etc.). Once the module is transmitting, then attempt to communicate with the slave, then enter any other

An example of a command list is shown below. Note that the commands can be entered in rows and that once the column definitions are understood, reviewing the Command List is very easy.

	0 PORT NUM	1 SLV ADD	2 FUNC CODE	3 SRC ADD	4 CNT	5 DEST ADDR	6 TYPE	7 POLL TIME
N7:50	1	3	1	0	10	100	0	0
N7:60	1	3	4	50	20	100	0	0
N7:70	2	2	16	200	10	137	0	6
N7:80	6	2	16	210	10	150	0	0

Example Command List

An example of multiple message configuration data blocks is shown in the following table (Columns 8-9 not shown for clarity).

4.5 Command Control Mode - Master Mode

Under some special operating conditions, it may be necessary for the ladder logic to be able to closely coordinate and control the execution of commands in the Command List. To accommodate this requirement, the MCM module supports something called the Command Control Mode.

When configured in the Command Control Mode, the ladder logic is able to provide Command Enable control on a per Command List entry basis. In addition, when used in conjunction with the Command Done Bits (See Section 5.3), the ladder logic is able to effectively one-shot each command if desired.

4.5.1 The BTW Block Structure

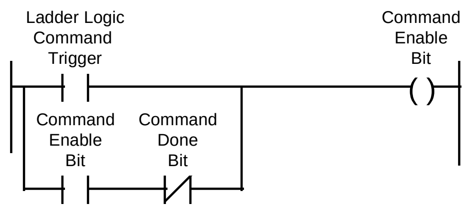
The structure of the Enable bits as they are written to the module in the BTW Block Transfer buffer is as follows:

Word	Name	Description
0	BTW Block ID	The Command Enable bits are moved to the module during every BTW transaction. Therefore, all valid BTW Block ID numbers can be used here
1-50	Data	Module data and Command List, as outlined above
51-56	Cmd Enable Bits	These registers contain Command Enable Bits for each command in the command list, up to the first 96 commands. The Enable Bits are bit mapped into the words depending on their relative position in the Command List. The mapping within the words is as follows: **Word** **Cmds** 51 1 to 16 52 17 to 32 53 33 to 48 54 49 to 64 55 65 to 80 56 81 to 96 Example : Word 51 bit 0 is Command #1 Enable

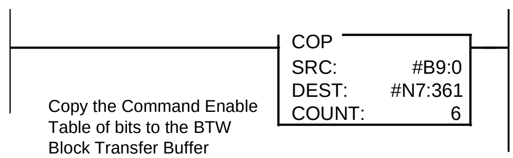
4.5.2 Controlling the Commands

When a command is configured in the Command Control Mode, and when the module detects the Command Enable bit changing state from 0 to 1, the module will attempt to execute the command (Three attempts will be made to execute the command). If the command is successfully sent, the Command Done bit will be set. If an error occurs during the sending process, the Command Error bit will be set.

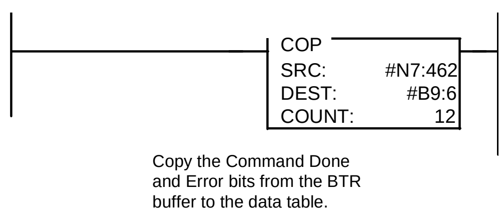
As example of the ladder logic which might be implemented to control a command would appear in structure as follows:



The simplest implementation would be to maintain a Binary table of Command Enable Bits which is copied to the BTW Buffer every transaction. The following branch of logic can be added to the BTW rung (transfer data to module):



The Command Done and Error bits could then be copied into the same Binary File and referenced in ladder logic after being transferred. The following instruction can be added to the BTR rung (read data from module) accomplish this:



4.5.3 Example Command List

Commands can be controlled through configuration of the Command Enable

	0 PORT NUM	1 SLV ADD	2 FUNC CODE	3 SRC ADD	4 CNT	5 DEST ADDR	6 TYPE	7 POLL TIME
N7:50	9	3	1	0	10	100	0	0
N7:60	10	3	4	50	20	100	0	0

Example Command List
An example where the command in N7:50 is configured as a Control Command Mode for Port 1 while the N7:60 command is configured for Port 2.

4.6 Event Initiated Commands - Master Mode [BTW Block ID Codes 100 to 119]

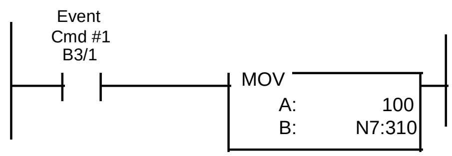
In addition to the continuously enabled commands which can be configured in the Command List, the MCM module also supports Event Initiated commands. These Event Commands can be used for reading/writing data conditionally with a slave. Example applications might include setting the time in a slave, resetting a batch counter, etc.

Hints to Make Life Easier

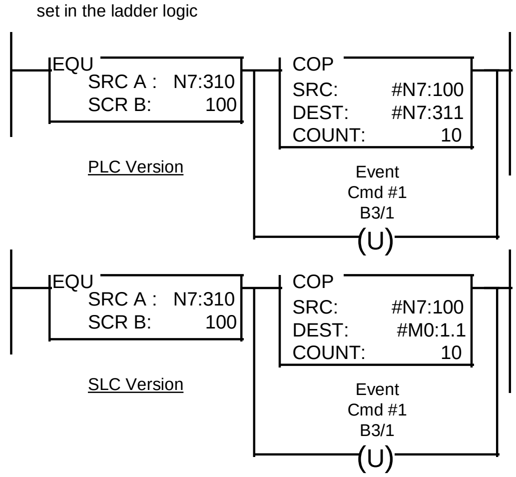
One of the benefits offered by an Event Initiated Write Command (FC 5, 6, 15, 16) is that the data contents written to the slave are guaranteed to be coordinated with the execution of the command. Note that this is not necessarily the case when executing the commands out of the Command List.

4.6.1 Ladder Logic

Executing an Event Initiated Command is performed by transferring data to the BTW Buffer while the Block ID number is between 100 and 119, inclusive. The data block which is transferred, detailed in the next section, contains the data necessary for the module to encode a valid command The additions to the ladder logic which must be made to support this functionality are as follows:



This branch is added to the Read rung, just above the MOV 255 to N7:310 branch. The B3/1 bit, selected here for example purposes only, is one-shot set in the ladder logic



This branch is added to the BTW rung, and serves to copy the Event Initiated Command block structure to the module and then Unlatches the command enable bit which was set in the ladder program.

4.6.2 BTW Block Structure

The structure of the block containing the Event Initiated Command is shown in the table below:

See Section 6 for details on configuring Modbus Commands

BTW Word	Data Offset	Name	Description
0		BTW Block ID	The BTW Block ID is used by the module to determine that the Block Transfer buffer contains an Event Initiated Command. Valid values are between 100 and 119. The value determines the relative position in the Master Error Table.

(Cont'd)

BTW Word	Data Offset	Name	Description
1	N[ ]:0	Port Number	The Port Number parameter allows the application to select which port the module will use to execute the command. Expected values are: **Port/Mode** — **Description** 1 — Port 1 2 — Port 2
2	N[ ]:1	Slave Address	The slave address represents the Modbus slave address of the slave station to which the command is directed. Addresses should be entered in the decimal form.
3	N[ ]:2	Function Code	The function code entered in the table tells the MCM Module what command to execute. The different choices are detailed in Section 6, but in an overview they are as follows: **Function Code** — **Description** 1 — Read Output Status 2 — Read Input Status 3 — Read Multiple Data Registers 4 — Read Input Registers 5 — Force Single Coil (Latch/Unlatch) 6 — Preset (Write) Single Data Register 15 — Multiple Coil Latch/Unlatch 16 — Preset (Write) Multiple Data Register
4	N[ ]:3	Source Address	The value represents the register or bit address, for read commands, from which data will be obtained. - When issuing a read command, the Source Register Address is the register location in the slave where the command will begin getting data When executing an Event Initiated Write command, this value has no meaning
5	N[ ]:4	Count	The number of words or bits the Modbus command is to read or write. See Section V for a detailed discussion on the word and bit lengths to be specified for the different commands.
6	N[ ]:5	Destination Address	The value represents the register or bit address, for both read and write commands, to which data will be written. The distinction between the two is as follows: - When issuing a read command, the Destination Address is the register location in the module where the command will begin placing the data from the slave - When issuing a write command, the Destination Address is register in the slave where the command will begin placing the data to be written to the slave.
7	N[ ]:6	Type	The Type field is relevant only during a Function Code 3 command (Multiple Register Read). The Type field tells the module to execute word swapping on the data being received by that particular command. See Section 4.4.2 for a complete discussion of this parameter
8-51	N[ ]:7	Write Data	These register contain the write data values which will be sent to the address slave per the Function Code selection.

	0 PORT NUM	1 SLV ADD	2 FUNC CODE	3 SRC ADD	4 CNT	5 DEST ADDR	6 TYPE	7 DATA	8 DATA	9 DATA
N7:100	1	3	1	0	10	100	0	0	0	0
N7:110	1	3	6	0	1	1	0	1267		

Example Event Initiated Write Commands

The first command issues a FC 1 to Slave 3, reading 10 bits from bit 0 into register 100 in the module. The second commands writes the value 1267 to register 1 in the slave.

5 Reading from the Module

This section provides reference level details on the transfer of data from the PLC/SLC processor to the MCM module. This type of transfer allows the ladder logic to send configuration , command list and data to the module.

5.1 Transferring data from the module [BTR Block ID 0 to 79]

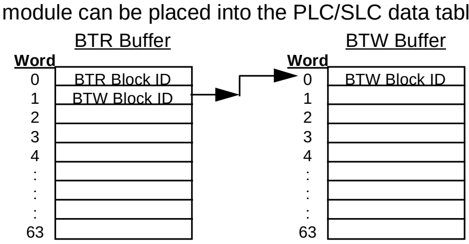
When the Master port driver reads data from a slave or when a Host writes to the Slave port driver, the resulting data is placed into the ProSoft module's data space (Addresses 0 to 3999). This Module Data space is the same block of memory that the PLC/SLC can write into per the above discussion.

The transfer of data from the ProSoft Technology module to the processor is executed through the Block Transfer Read function. The following sections detail the handling of the read data.

> Although the full physical 64 words of the data buffer may not be used, the BTR and M1 lengths must be configured for a length of 64 words, otherwise module operation will be unpredictable

5.1.1 The Read Data Block Structure

The BTR buffer definition is:

Word	Name	Description
0	BTR Block ID	The ladder logic uses this value to determine the contents of the data portion of the BTR buffer. With some conditional testing in ladder logic, the data from the module can be placed into the PLC/SLC data table.  The relationship between the BTR Block ID number and the register table can be put into an equation: Starting Register Address = Block ID Number * 50 Valid codes are between 0 and 79.
1	BTW Block ID	The module returns this value to the processor to be used to enable the movement of register data and command list blocks to the module. The BTW Block ID number is developed by the module based on the parameters entered in parameters 21 and 22 of Block 255. This value is intended to only be a suggestion and to ease the ladder logic programming requirements. If it is desired to develop a different data transfer series, this may be easily accomplished in ladder logic. Valid codes are: **BTW Code** — **Description** 0-79 — Module Data 80-99 — Command List 100-119 — Event Initiated Cmds 255 — Module Configuration

(Cont'd)

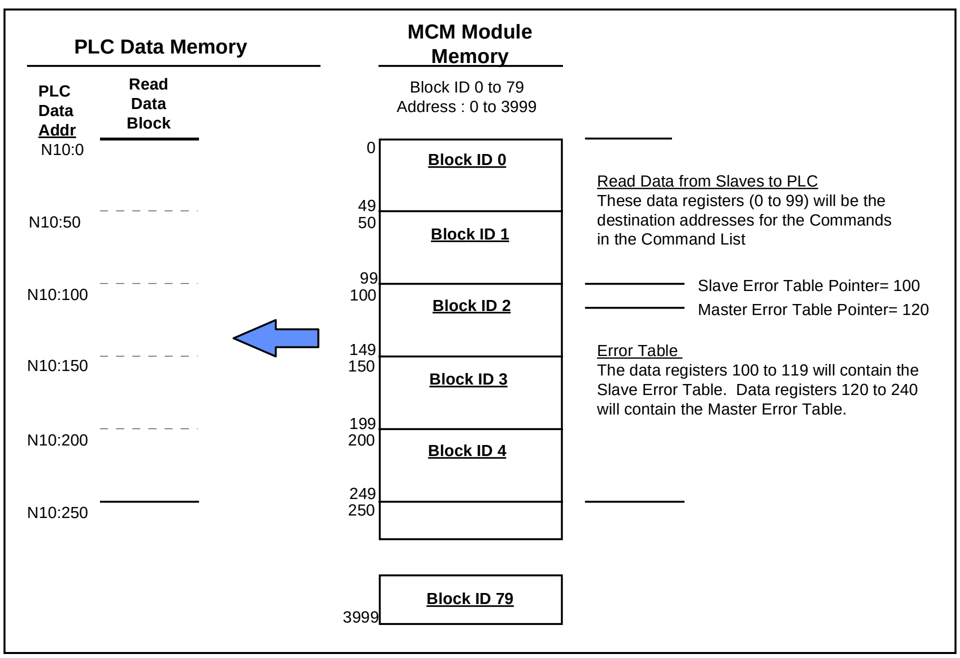
2 to 51	Data	The contents of the module's Register Data space (0 - 3999). This data will contain data received from the slaves, data moved from the processor, and the Slave and Master Error Tables. The values will be 16 bit register values, and should be placed into integer files. Note that the user application ladder logic controls the placement and use of the data registers.
52 to 63	Command Done and Error Bits	See Section 5.3

5.1.2 Moving the data from the module to the processor

Data which has been read from the slave devices (Master driver) or has been written from a host (Slave driver) is deposited into a 4000 word register table in the module. This table is addressed starting at 0 and going up to 3999.

The data register table is transferred from the module to the ladder logic through a paging mechanism designed to overcome the 64 physical word limit of the BTR instruction. The paging mechanism is outlined in the discussion above, but the important thing to understand is the relationship between the page numbers (BTR Block ID numbers) and the register addresses in the module.

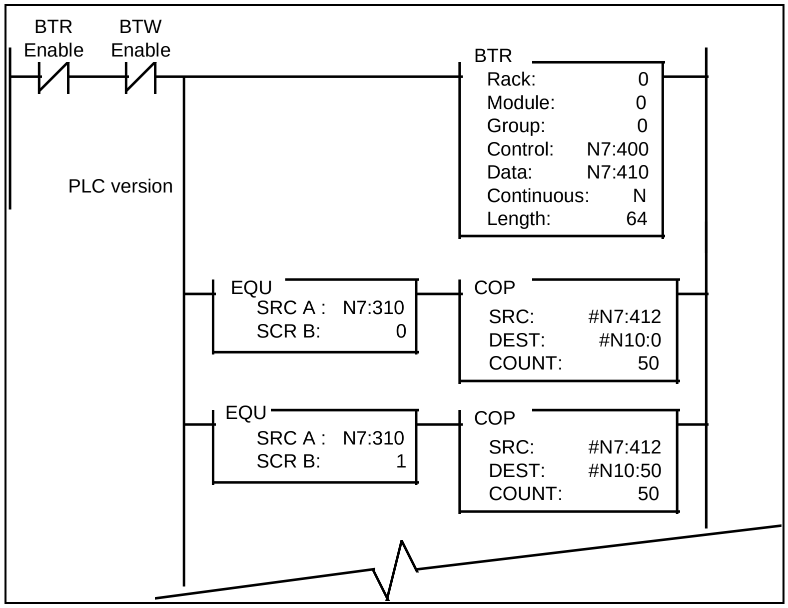
The diagram also shows the layout for an example application. Note the number of blocks returned from the module to the ladder logic is determined by the value entered in the System Configuration 'Read Block Cnt' register . In this example we have assumed a Read Block Count value of 5.



Read Data Blocks from MCM Module
Note that this diagram assumes a Read Block Count value of 5, therefore returning Registers 0 to 249 from the module. This value can be altered as needed depending on the application.

5.1.3 Ladder Logic to Read Module Data

The ladder logic must be programmed to look at the BTR buffer, decode several words, and then take action. The following is an example of such ladder logic:



Example ladder to transfer data from module

This logic shows a method for moving data from the module to the PLC data table.

5.1.4 Slave Error Code Table

The MCM Module monitors the status of all Slave port commands. This status is communicated to the processor in the form of a Slave Error Code Table.

> The Slave Error Code Table is initialized to zero on power up, and every time the module receives the 255 configuration data block.

The Slave Error Table is a 20 word block. The location of the Error Table is determined by the Slave Error Table Pointer parameter in the Configuration Block. The structure of the data block is as follows:

Port 1 Status Codes

Word	Example Addr	Name	Description
0	N10:100	Current Port Status	This value represents the current value of the error code for the port. This value will only be valid if the port is configured as a Slave. The possible values are detailed in the following section.
1	N10:101	Last Transmitted Error	This value is the last error code transmitted to the master by this slave port. Error codes which can be expected in this field are 0, 1, 2, 3, and 6. The field will only be cleared by re configuring the module (Block ID 255).
2	N10:102	Total Msgs to this slave	This value represents the total number of messages that have matched this slaves address on this port, whether the slave actually determined them to be good (worthy of response) or not.

Port 1 Status Codes (Cont'd)

3	N10:103	Total Responses from this slave	This value represents the number of good (non-error) responses that the slave has sent to the master on this port. The presumption is that if the slave is responding, the message was good.
4	N10:104	Total Msgs seen by this slave	This value represents the total number of commands seen by the slave on this port, regardless of the slave address.

Port 2 Status Codes

Word	Example Addr	Name	Description
5	N10:105	Current Port Status	Reference above for these descriptions
6	N10:106	Last Transmitted Error	Reference above for these descriptions
7	N10:107	Total Msgs to this slave	Reference above for these descriptions
8	N10:108	Total Responses from this slave	Reference above for these descriptions
9	N10:109	Total Msgs seen by this slave	Reference above for these descriptions

System Information

Word	Example Addr	Name	Description
10-11	N10:110 N10:111	Product Name (ASCII)	These two words represent the product name of the module in an ASCII representation. In the case of the MCM product, the letters ' MCM ' should be displayed when placing the programming software in the ASCII data representation mode.
12-13	N10:112 N10:113	Revision (ASCII)	These two words represent the product revision level of the firmware in an ASCII representation. An example of the data displayed would be '1.45' when placing the programming software in the ASCII data representation mode.
14-15	N10:114 N10:115	Operating System Rev (ASCII)	These two words represent the module's internal operating system revision level in an ASCII representation.
16-17	N10:116 N10:117	Production Run Number (ASCII)	This number represents the 'batch' number that your particular chip belongs to in an ASCII representation.
18-19	N10:118 N10:119	Spare	

All counters in the Slave Error Table will rollover to 0 after reaching 65535

5.1.5 Master Error Code Table

The MCM Module monitors the status of all Master port commands. This status is communicated to the processor in the form of a Master Error Code Table, the position of which is controlled by the Master Error Table Pointer in the Communication Configuration setup. Each Master command will generate an Error Code for use by the user.

The Master Error Code Table is initialized to zero on power up, and every time the module receives the 255 configuration data block.

The Error Code Table is a 120 word block. The relationship between the placement of the error codes within the Error Table and the commands is according to the command's relative position in the command list.

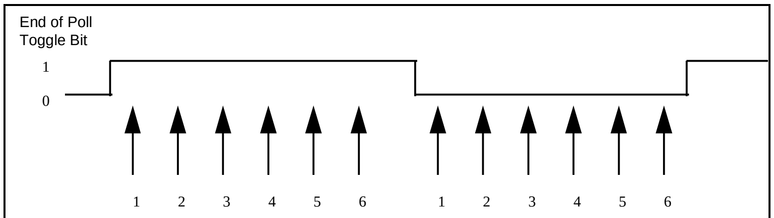
The simplest method for obtaining the Master Error Status Table is to locate it at the end of the application's data map and then read it back into the PLC/SLC data table as part of the regular data. The structure of the Master Error Table is as follows:

Word	**Description**
0	Command List End of Poll Status
1	Command #1 Error Status
2	Command #2 Error Status
-	
98	Command #98 Error Status
99	Command #99 Error Status
100-120	Future

Where:

Command List End Of Poll Status: This register provides an indication of when the Master has completed one cycle through the Command List. A bit in the word will be toggled each time the command list has been completed. The status is indicated for each master port as follows:

Bit	
0	Master Port 1
1	Master Port 2



The theoretical operation of the End of Poll Toggle Bit is that all of the commands which are to be executed for a port are execute within each state change of the bit.

Command Error Status: The Error Status Codes, either received from the slaves, or generated by the module, are placed in the table. See the next section for the meaning of the error codes. The values will be 16 bit values, and should be placed into an integer file. Note that the user application ladder logic controls the placement and use of these registers.

Error Status Table Example
Master Error Table Pointer = 120

	Wrd 0	**Wrd 1**	**Wrd 2**	**Wrd 3**	**Wrd 4**	**Wrd 5**	**Wrd 6**	**Wrd 7**	**Wrd 8**	**Wrd 9**
N10:120	0	0	**8**	0	0	0	0	0	0	0
N10:130	0	0	0	0	0	0	0	0	0	0
N10:140	0	0	0	0	0	0	0	0	0	0
N10:150	0	0	0	0	0	0	0	0	0	0
N10:160	0	0	0	0	0	0	0	0	0	0
N10:170	0	0	0	0	0	0	0	0	0	0
N10:180	0	0	0	0	0	0	0	0	0	0
N10:190	0	0	0	0	0	0	0	0	0	0
N10:200	0	0	0	0	0	0	0	0	0	0
N10:210	0	0	0	0	0	0	0	0	0	0
N10:220	0	0	0	0	0	0	0	0	0	0
N10:230	0	0	0	0	0	0	0	0	0	0

These registers correspond to the registers used in the sample program for PLC-5 in the back of this manual. Your application may require your own specific program. In this case an error code of 8 was generated for command 2 -- all other commands were executed without any errors. Column 0 is used to identify that a master port has reached the end of the command list, and is starting at the top of the Command List

5.1.6 Error Status Codes

The Error Codes returned in the Slave and Master Error Code Tables reflect the outcome of the commands and responses executed by the module. Note that in all cases, if a zero is returned, there was not an error. Valid Error Status Codes are as follows:

Code	Name	Description
0	All OK	The module is operating as desired
1	Illegal Function	An illegal function code request is being attempted
2	Bad Data Address	The address, or the range of addresses, covered by a request from the master are not within allowed limits
3	Bad Data Value	The value in the data field of the command is not allowed.
4	Incomplete Response Detected	This error indicates that an incomplete response was received to a master query. Often this will indicate that the slave device may be responding too quickly or that there may be excessive noise on the line.
6	Module Busy	The module busy status code is returned when a write command from the master has not yet been completed when a second write command is received
8	Timeout Error	Communications with the addressed slave have been unsuccessful due to a lack of response from the slave. The Master port will attempt a command three times before moving onto the next command.
10	Buffer Overflow	The receive buffer has overflowed and reset the character count to 0. If this condition occurs try reading fewer parameters at one time
16	Port Configuration Error	If this value is returned from the module, one or both of the serial ports have been mis-configured. To determine the exact source of the problem, verify the following: - Parity Configuration - Stop Bit Configuration - Baud Rate Configuration - Start Input Register Address - Start Output Register Address
18	System Configuration Error	If this error is returned from the module, one of the system configuration parameters has been detected out of range. To determine the source, verify the following: - Read Block Count <= 80 - Write Block Count <=80 - Command Block Count <= 20 - Slave Error Pointer <= 3850 - Master Error Pointer <= 3880
254	Checksum Error	The slave determined that the message checksum was in error, and therefore discarded the message
255	TX Hardware Timeout	A transmit timeout condition has occurred indicating that the module was not able to transmit the command. Verify that the RTS-CTS jumper on the port is still connected

5.2 Pass-Through Mode - Slave Mode [BTR Block ID 256 to 259]

When a Slave port is configured to support the Pass-Through mode, any Modbus write commands which are addressed to the local slave address will be passed across the backplane for processing by the ladder logic. Ladder logic in the Appendix provides an example of how the Pass-Through commands can be decoded.

5.2.1 The Block Structure

The BTR buffer definition for Pass-Through Mode transfers is:

Word	Name	Description
0	BTR Block ID	The value of the BTR Block ID register represents the type of write command which has been received from the host. Valid codes are: **<u>BTR ID</u>** **<u>Description</u>** 256 Register Write 257 Register Write - Enron Float 258 Single Bit Write 259 Multiple Bit Write
1	BTW Block ID	Same as above description
2-62	Data	The contents of these registers are a function of the BTR Block ID number (i.e., the command received from the host)

5.2.2 Receiving Register Writes [BTR Block ID 256 and 257]

The BTR buffer definition for is:

Word	Name	Description
0	BTR Block ID	The value of the BTR Block ID register represents the type of write command which has been received from the host. Valid codes are: **<u>BTR ID</u>** **<u>Description</u>** 256 Register Write 257 Register Write - Enron Float 258 Single Bit Write 259 Multiple Bit Write
1	BTW Block ID	Same as above description
2	Count	The number of registers being written by the Master. Valid numbers which will be received will range from 1 to 60
3	Destination Address	This value is used by the ladder logic to determine the address in the processor data table in which to start the data write
4-62	Data	The data values written from the master. The value will be 16 bit register value

5.2.3 Receiving Single Bit Writes [BTR Block ID 258]

The BTR buffer definition is:

Word	Name	Description
0	BTR Block ID	The value of the BTR Block ID register represents the type of write command which has been received from the host. Valid codes are: **<u>BTR ID</u>** **<u>Description</u>** 258 Single Bit Write
1	BTW Block ID	Same as above description
2	Bit Address	Represents the bit which will be acted on
3	Control Action	The action being commanded by the master. When the value is 0, the addressed bit is to be reset and when the value is 1 the addressed bit is set

5.2.4 Receiving Multiple Bit Writes [BTR Block ID 259]

The BTR buffer definition is:

Word	Name	Description
0	BTR Block ID	The value of the BTR Block ID register represents the type of write command which has been received from the host. Valid codes are: **BTR ID** — **Description** 259 — Multiple Bit Write
1	BTW Block ID	Same as above description
2	Word Count	Represents the number of words in the data block that contain valid bit data. Valid numbers range from 1 to 30 (This limits the number of bits which can be written in one command to 30 * 16).
3	Word Start Address	Represents the offset word address into which the bit write data block will start to be written. When the master addresses a bit write, it sends the starting bit address. The starting bit address is used by the module to generate this word start address (Bit address/ 16)
4-33	Data	These registers contain the bit write data received from the master. Note that partial work length bit writes are acceptable. The mask bits and some ladder logic protects un-addressed bits within a common word.
34-63	Mask	These words mask off the un-addressed bits. This allows for starting addresses which are not on a word boundary and lengths which do not end on a word boundary. Reference the example ladder logic in the Appendix.

5.3 Decoding Command Done and Command Error Bits - Master Mode

The Command Done and Command Error bits are returned for use in the ladder logic program during every data block transfer (BTR Block ID 0 to 79). These bits can be used by the ladder logic to keep track of command execution or to disable commands when a command is configured in the Command Control Mode (See Section 4.5).

5.3.1 The Block Structure

The structure of the Done and Error bits as they are returned in the BTR Block Transfer buffer is as follows:

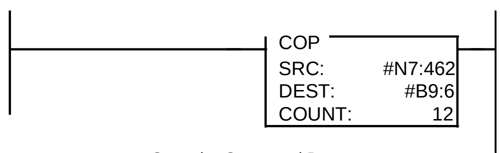
Word	Name	Description
0	BTR Block ID	When the BTR Block ID value is between 0 and 79 the BT Buffer contains Command Done and Command Error bits, as outlined below
1	BTW Block ID	Same as above description
2-51	Data	Module data, as outlined above
52-57	Cmd Done Bits	These registers contain Done Bit flags for each command in the command list, up to the first 96 commands. The Done Bits are bit mapped into the words depending on their relative position in the Command List. The mapping within the Done Bits is as follows: **Word** — **Cmds** 52 — 1 to 16 53 — 17 to 32 54 — 33 to 48 55 — 49 to 64 56 — 65 to 80 57 — 81 to 96 Example : Word 52 bit 0 is Command #1

(Cont'd)

Word	Name	Description
58-63	Cmd Error Bits	These registers contain Error Bit flags for each command in the command list, up to the first 96 commands. The Error Bits are bit mapped into the words depending on their relative position in the Command List. The mapping within the Done Bits is as follows: **Word** **Cmds** 58 1 to 16 59 17 to 32 60 33 to 48 61 49 to 64 62 65 to 80 63 81 to 96 Example : Word 52 bit 0 is Command #1

5.3.1 Ladder Logic

A simple rung of logic can be entered to move the Done and Error bits from the BTR buffer to the PLC/SLC data table. An example follows:



Copy the Command Done and Error bits from the BTR buffer to the data table.

6 Modbus Command Configuration

The ProSoft Technology MCM Modbus Master and Slave communication drivers support several data read and write commands. When configuring a Master port, the decision on which command to use is made depending on the type of data being addressed, and the level of Modbus support in the slave equipment. When configuring as a slave, it may be important to understand how the Modbus commands function in order to determine how to structure the application data.

We have included an excerpt from the Modbus Protocol Specification in the Appendix to assist in thoroughly understanding the functionality of the Modbus protocol.

6.1 Modbus Commands

The MCM module supports a command subset of the Modbus Specification consisting primarily of the Function Codes required to read and write data. The following sections detail the different commands supported by the module. The perspective is given mainly from that of a Master port, but much of the discussion will assist those implementing Slave ports.

Function Code	Cmd	Address Range	Slave Driver Comments	Master Driver Comments
1	Read Output Status	Coil 0001 to 9999	Module returns binary data from the 'Output Status' register space. The module will support up to 125 words of data in one command.	Source Addr : Starting bit address in the slave from which data should be read. Enter a 0 to address coil 0001 Count :Number of bits to be read (up to 125 words in length) Dest Addr : Starting word address in the module's Register Memory in which the data should be placed, starting with bit 0.
2	Read Input Status	Bit 10001 to 29999	Module returns binary data from the 'Input Status' register space. The module will support up to 125 words of data in one command.	Source Addr : Starting bit address in the slave from which data should be read. Enter a 0 to address bit 10001 Count :Number of bits to be read (up to 125 words in length) Dest Addr : Starting word address in the module's Register Memory in which the data should be placed, starting with bit 0.

(Cont'd)

Function Code	Cmd	Address Range	Slave Driver Comments	Master Driver Comments
3	Read Multiple Registers	Registers 40001 to 47999	The module returns word data from the register space. The entire 4000 words in the module make up the register space addressable by a host. Word 0 in the module corresponds to Modbus Address 40001. The module will support up to 125 words of data in one command.	Source Addr : Starting register address in the slave from which data should be read. Enter a 0 to address 40001 in the slave Count :Number of words or values to be read (up to 125 words in length) Dest Addr : Starting word address in the module's Register Memory in which the data should be placed. Type : Controls byte and word swapping for floating point read (See Section 4 for details).
4	Read Input Registers	Registers 30001 to 39999	The module returns data from the 'Input Register' space in the module. The module will support up to 125 words of data in one command.	Source Addr : Starting register address in the slave from which data should be read. Enter a 0 to address 30001 in the slave Count :Number of words or values to be read (up to 125 words in length) Dest Addr : Starting word address in the module's Register Memory in which the data should be placed.
5	Single Bit/Coil Write	Coil 0001 to	Normal Mode : The bit written will be placed in the module's data space Pass-Through Mode : The bit written will be passed to the PLC/SLC for handling in ladder logic	Source Addr : Starting bit address in the MCM which should be used to determine the bit set/reset action of the command Count :Not used, defaults to one Dest Addr : The bit address in the slave which is to be set or reset. Enter an address of 0 to address coil 0001 in the slave
6	Single Register Write	Registers 40001 to 47999	Normal Mode : The bit written will be placed in the module's data space Pass-Through Mode : The bit written will be passed to the PLC/SLC for handling in ladder logic	Source Addr : Starting register address in the MCM which should be used to determine the source of the data to be written Count : Not used, defaults to one Dest Addr : The register address in the slave in which the data is to be written. Enter an address of 0 to address register 40001 in the slave

(Cont'd)

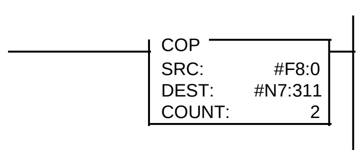
15	Multiple Bit/Coil Write		Normal Mode : The bit written will be placed in the module's data space. Up to 125 words of bit data max. Pass-Through Mode : The bit written will be passed to the PLC/SLC for handling in ladder logic. Up to 30 words of bit data max.	Source Addr : Starting bit address in the MCM which should be used to determine the source of the data to be written Count :The number of bits to be written (up to 125 words total) Dest Addr : The starting bit address in the slave in which the data is to be written. Enter an address of 0 to address coil 0001 in the slave
16	Multiple Register Write		Normal Mode : The register value written will be placed in the module's data space. Up to 125 words max Pass-Through Mode : The bit written will be passed to the PLC/SLC for handling in ladder logic. Up to 60 words max.	Source Addr : Starting register address in the MCM which should be used to determine the source of the data to be written Count :The number of words or values to be written (up to 125 words total) Dest Addr : The starting address in the slave in which the data is to be written. Enter an address of 0 to address register 40001 in the slave

6.2 Floating Point Support

The movement of floating point data between the MCM module and other devices is easily accomplished as long as the device supports IEEE 754 Floating Point format. This IEEE format is a 32-bit single precision floating point format.

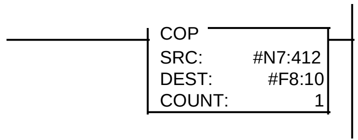
The programming necessary to move the floating point data is to take advantage of the COP command that exists in the PLC and the SLC. The COP command is unique in the PLC/SLC data movement commands in that it is an untyped function, meaning that no data conversion is done when moving data between file types (i.e., it is an image copy not a value copy).

The structure of the COP command to move data from a Floating Point file into an integer file (something you would do to move floating point values to the module) is as follows:



This command will move one floating point value in two 16 bit integer images to the integer file. For multiple floating point values simply increase the count field by a factor of 2 per floating point value.

The structure of the COP command to move data from an Integer file to a Floating Point file (something you would do to receive floating point values from the module) is as follows:



This command will move two 16 bit integer registers containing one floating point value image to the floating point file. For multiple values simply increase the count field.

6.2.1 ENRON Floating Point Support

Many manufacturers have implemented special support in their drivers to support what is commonly called the Enron version of the Modbus protocol. In this implementation, register addresses > 7000 are presumed to be floating point values. The significance to this is that the count field now becomes a 'number of values' field. In floating point format, each value represents two words. This have been implemented in the MCM in the following manner:

Function Code	Master Driver	Slave Driver
3 - Register Read	If the Destination Address is >= 7000 then expect 2 words per Count value requested.	If the register addressed by the host is >= 7000 then the following equation is used to determine the MCM register address to be read from: MCM Addr = float_offset + (reg_addr - 7000) * 2
6/16 - Register Write	If the Destination Address is >= 7000 then send two words per Count value (i.e., Count value becomes number of values not number of words)	If the register addressed by the host is >= 7000 then the following equation is used to determine the MCM register address to be written into: MCM Addr = float_offset + (reg_addr - 7000) * 2

7 Diagnostics and Troubleshooting

Several hardware diagnostics capabilities have been implemented using the LED indicator lights on the front of the module. The following sections explain the meaning of the individual LEDs for both the PLC and the SLC platforms.

7.1 3100 PLC Platform LED Indicators

The PLC platform MCM product is based on the ProSoft CIM hardware platform. The following table documents the LEDs on the 3100-MCM hardware and explains the operation of the LEDs.

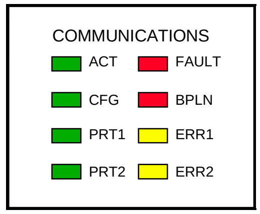
ProSoft CIM
Card

ACTIVE	O	O	FLT
CFG	O	O	BPLN
ERR1	O	O	ERR2
TXD1	O	O	TXD2
RXD1	O	O	RXD2

ProSoft CIM	Color	Status	Indication
ACT	Green	Blink (Fast)	Normal state : The module is operating normally and successfully Block Transferring with the PLC
		On	The module is receiving power from the backplane, but there may be some other problem
		Off	The module is attempting to Block Transfer with the PLC and has failed. The PLC may be in the PGM mode or may be faulted
FLT	Red	Off	Normal State : No system problems are detected during background diagnostics
		On	A system problem was detected during background diagnostics. Please contact factory for technical support
CFG	Green	Off	Normal state : No configuration related activity is occurring at this time
		Blink	This light blinks every time a Module Configuration block (ID = 255) is received from the processor ladder logic
		On	The light is on continuously whenever a configuration error is detected. The error could be in the Port Configuration data or in the System Configuration data. See Section 4 for details
BPLN	Red	Off	Normal State : When this light is off and the ACT light is blinking quickly, the module is actively Block Transferring data with the PLC
		On	Indicates that Block Transfers between the PLC and the module have failed.(Not activated in the initial release of the product)
ERR1 ERR2	Amber	Off	Normal State : When the error LED is off and the related port is actively transferring data, there are no communication errors
		Blink	Periodic communication errors are occurring during data communications. See Section 4 to determine the error condition
		On	This LED will stay on under several conditions: • CTS input is not being satisfied • Port Configuration Error • System Configuration Error • Unsuccessful comm on MCM slave Recurring error condition on MCM master
Tx1 Tx2	Green	Blink	The port is transmitting data.
Rx1 Rx2	Green	Blink	The port is receiving data

7.2 3150 SLC Platform LED Indicators

The following table documents the LEDs on the 3150-MCM hardware and explains the operation of the LEDs.



LED Name	Color	Status	Indication
ACT	Green	Blink (Fast)	Normal state : The module is operating normally and successfully Block Transferring with the SLC
		On	The module is receiving power from the backplane, but there may be some other problem
		Off	The module is attempting to Block Transfer with the SLC and has failed. The SLC may be in the PGM mode or may be faulted
FLT	Red	Off	Normal State : No system problems are detected during background diagnostics
		On	A system problem was detected during background diagnostics. Please contact factory for technical support
CFG	Green	Off	Normal state : No configuration related activity is occurring at this time
		Blink	This light blinks every time a Module Configuration block (ID = 255) is received from the processor ladder logic
		On	The light is on continuously whenever a configuration error is detected. The error could be in the Port Configuration data or in the System Configuration data. See Section 4 for details
BPLN	Red	Off	Normal State : When this light is off and the ACT light is blinking quickly, the module is actively Block Transferring data with the SLC
		On	Indicates that Block Transfers between the SLC and the module have failed
ERR1 ERR2	Amber	Off	Normal State : When the error LED is off and the related port is actively transferring data, there are no communication errors
		Blink	Periodic communication errors are occurring during data communications. See Section 4 to determine the error condition
		On	This LED will stay on under several conditions: • CTS input is not being satisfied • Port Configuration Error • System Configuration Error • Unsuccessful comm on MCM slave • Recurring error condition on MCM master
PRT1 PRT2	Green	Blink	The port is communicating, either transmitting or receiving data

7.3 Troubleshooting - General

In order to assist in the troubleshooting of the module, the following tables have been put together to assist you. Please use the following to help in using the module, but if you have additional questions or problems please do not hesitate to contact us.

The entries in this section have been placed in the order in which the problems would most likely occur after powering up the module.

Problem Description	Steps to take
BPLN light is on (SLC)	The BPLN light comes on when the module does not think that the SLC is in the run mode (i.e., SLC is in PGM or is Faulted). If the SLC is running then verify the following: • Verify the SLC Status File to be sure the slot is enabled • The Transfer Enable/Done Bits (I/O Bits 0 for the slot with the module) must be controlled by the ladder logic. See Section 2.x for details or the example ladder logic in the Appendix. • If the ladder logic for the module is in a subroutine file verify that there is a JSR command calling the SBR
CFG light does not clear after power up (no ERR LED)	The 255 BTW Block ID number is not being detected by the module. This could be due to a Block Transfer failure (PLC) or to an error in the ladder logic preventing the 255 value from being moved to the BTW buffer
CFG light does not clear after power up (w/ ERR LED)	If the BPLN light has been cleared, then several of the Port and System configuration values are value checked by the module to be sure that legal entries have been entered in the data table. Verify the Error Status Table for an indication of a configuration error.
CFG light toggles	Under normal conditions, the CFG LED will clear immediately after receipt. If the CFG light toggles, this usually indicates that the logic condition which places the 255 Block ID value in the BTW buffer is not being cleared. Check the ladder logic to be sure that the condition moving the 255 value is not held true.
Module is not transmitting	Presuming that the processor is in run, verify the following: • CTS input is not satisfied (check RTS/CTS jumper) • Check Error Status codes for 255 code. If so see next problem • If in slave mode, verify the slave address being requested from the Host • If in master mode, verify the command list configuration and that the Command List is being moved into the module (i.e., check the Command Block Cnt and associated ladder logic)
Error Code 255 in Status Table	This is caused by only one thing, a missing CTS input on the port. If a cable is connected to the port, then verify that a jumper has been installed between the RTS and CTS pins. If so then there may be a hardware problem.
Overwriting data blocks	This condition normally occurs when it is forgotten that the BTW Block ID value is being manipulated by the module, and that it always starts at 0. Please verify that the configuration of the module (Read and Write Block Counts) is not causing data from the PLC/SLC to overwrite data being returned from the module. A simple method for verifying this is to perform a histogram on the BTW Block ID register.
Data swapping is occurring (3100 only)	Under several circumstances data swapping in the module has occurred. This swapping has always been associated with the 8/16 pt jumper on the back of the card. Please verify that the jumper is in the 8pt position
New configuration values are not being accepted by the module	In order for new values to be moved to the module a Block Transfer Write with a Block ID of 255 must be transmitted to the module. The 'User Config Bit' in the example logic accomplishes this. In the example logic the bit must either be set in the data table manually or the module must be powered down/reset. In order to download the configuration upon transitioning from PGM to RUN, simply add a run to set the 'User Config Bit' based on the First Scan Status Bit (S1:1/15)

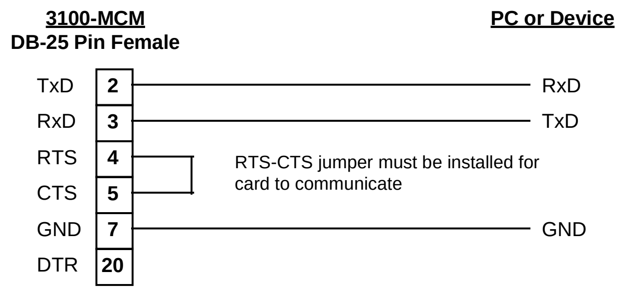
Problem Description	Steps to take
Error Codes being returned in locations with no commands (Master Configuration)	Be sure that the Command Block Count configuration value is setup correctly. There should be one branch of logic in the Write Rung corresponding to each Command Block to be written (i.e., a Command Block Count of 2 should have two branches of logic to handle BTW Block IDs 80 and 81. If the Command Block Count configuration value exceeds the number of branches in logic, the Command List is inadvertently being duplicated. To resolve the issue, either add more branches of logic or reduce the Command Block Count value to match the number of BTW logic branches.
RX1 or RX2 on continuously (3100 only)	The TX and RX LEDs on the module are tied to the hardware state of the ports (i.e., are not controlled directly by firmware). When the RX LED is on continuously is normally indicates that the polarity of the cable connection to the port is swapped. This is particularly true in RS-485 and RS-422 modes.

8 Cable Connection Diagrams

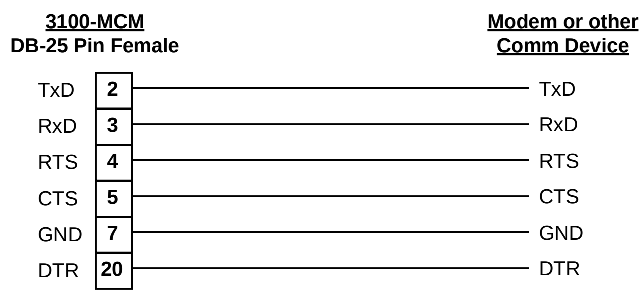
The following diagrams show the connection requirements for the ports on the 3100 and 3150 modules.

3100-MCM Module

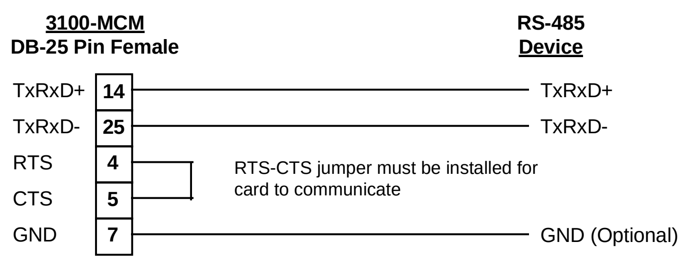
RS-232 w/ No Hardware Handshaking
Port Connection with another communication port



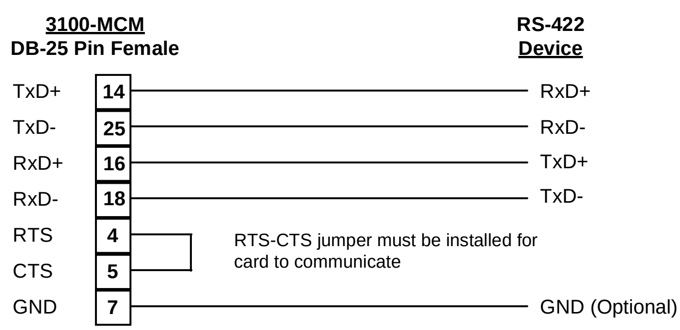
RS-232 w/ Hardware Handshaking
Port Connection with a modem or other similar device



RS-485/2-Wire Connection
The jumper on the module must be set in the RS-485 position for all 2-wire applications

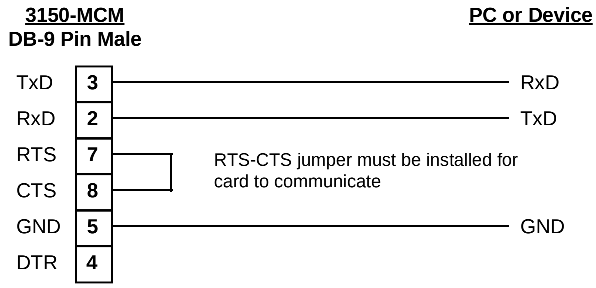


RS-422/4-Wire Connection
The jumper on the module must be in the RS-422 position for all 4-wire applications

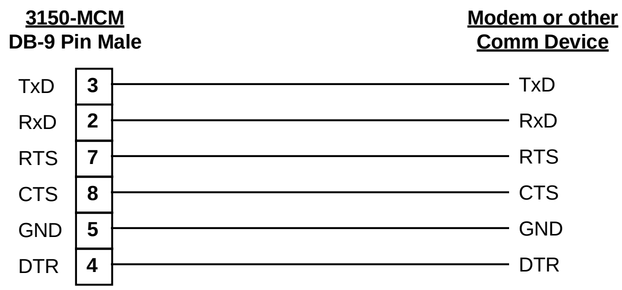


3150-MCM Module

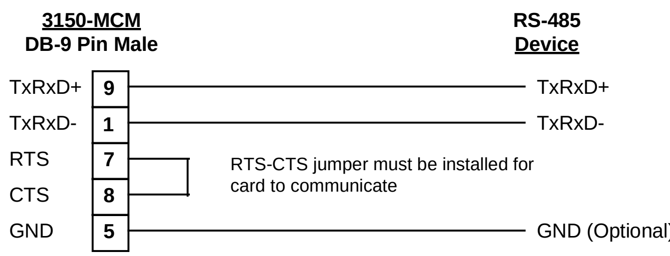
RS-232 w/ No Hardware <u>Handshaking</u>
Port Connection with another communication port



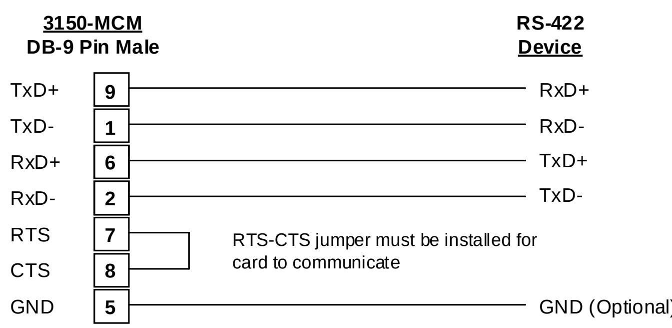
RS-232 w/ Hardware <u>Handshaking</u>
Port Connection with a modem or other similar device



<u>RS-485/2-Wire Connection</u>
The jumper on the module <u>must</u> be set in the RS-485 position for all 2-wire applications



<u>RS-422/4-Wire Connection</u>
The jumper on the module must be in the RS-422 position for all 4-wire applications



<u>RS-485 and RS-422 Tip</u>
If communication in the RS-422/RS-485 mode does not work at first, despite all attempts, try switching termination polarities. Some manufacturers interpret +/- and A/B polarities differently.

A Support, Service and Warranty

Technical Support

ProSoft Technology survives on its ability to provide meaningful support to its customers. Should any questions or problems arise, please feel free to contact us at:

Factory/Technical Support
ProSoft Technology, Inc.
9801 Camino Media, Suite 105
Bakersfield, CA 93311
(661) 664-7208
(800) 326-7066
(661) 664-7233 (fax)
E-mail address: prosoft@prosoft-technology.com
Web Site : http://www.prosoft-technology.com

Before calling for support, please prepare yourself for the call. In order to provide the best and quickest support possible, we will most likely ask for the following information (you may wish to fax it to us prior to calling):

1. Product Version Number
2. Configuration Information
   - Communication Configuration
   - Master Command List
   - Jumper positions
3. System hierarchy
4. Physical connection information
   - RS-232, 422 or 485
   - Cable configuration
5. Module Operation
   - Block Transfers operation
   - LED patterns

An after-hours answering system (on the Bakersfield number) allows pager access to one of our qualified technical and/or application support engineers at any time to answer the questions that are important to you.

Module Service and Repair

The MCM card is an electronic product, designed and manufactured to function under somewhat adverse conditions. As with any product, through age, misapplication, or any one of many possible problems, the card may require repair.

When purchased from ProSoft Technology, the module has a one year parts and labor warranty according to the limits specified in the warranty. Replacement and/or returns should be directed to the distributor from whom the product was purchased. If you need to return the card for repair, it is first necessary to obtain an RMA number from ProSoft Technology. Please call the factory for this number and display the number prominently on the outside of the shipping carton used to return the card.

General Warranty Policy

ProSoft Technology, Inc. (Hereinafter referred to as ProSoft) warrants that the Product shall conform to and perform in accordance with published technical specifications and the accompanying written materials, and shall be free of defects in materials and workmanship, for the period of time herein indicated, such warranty period commencing upon receipt of the Product.

This warranty is limited to the repair and/or replacement, at ProSoft's election, of defective or non-conforming Product, and ProSoft shall not be responsible for the failure of the Product to perform specified functions, or any other non-conformance caused by or attributable to: (a) any misapplication of misuse of the Product; (b) failure of Customer to adhere to any of ProSoft's specifications or instructions; (c) neglect of, abuse of, or accident to, the Product; or (d) any associated or complementary equipment or software not furnished by ProSoft.

Limited warranty service may be obtained by delivering the Product to ProSoft and providing proof of purchase or receipt date. Customer agrees to insure the Product or assume the risk of loss or damage in transit, to prepay shipping charges to ProSoft, and to use the original shipping container or equivalent. Contact ProSoft Customer Service for further information.

Limitation of Liability

EXCEPT AS EXPRESSLY PROVIDED HEREIN, PROSOFT MAKES NO WARRANT OF ANY KIND, EXPRESSED OR IMPLIED, WITH RESPECT TO ANY EQUIPMENT, PARTS OR SERVICES PROVIDED PURSUANT TO THIS AGREEMENT, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANT ABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER PROSOFT OR ITS DEALER SHALL BE LIABLE FOR ANY OTHER DAMAGES, INCLUDING BUT NOT LIMITED TO DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION IN CONTRACT OR TORT (INCLUDING NEGLIGENCE AND STRICT LIABILITY), SUCH AS, BUT NOT LIMITED TO, LOSS OF ANTICIPATED PROFITS OR BENEFITS RESULTING FROM, OR ARISING OUT OF, OR IN CONNECTION WITH THE USE OR FURNISHING OF EQUIPMENT, PARTS OR SERVICES HEREUNDER OR THE PERFORMANCE, USE OR INABILITY TO USE THE SAME, EVEN IF PROSOFT OR ITS DEALER'S TOTAL LIABILITY EXCEED THE PRICE PAID FOR THE PRODUCT.

Where directed by State Law, some of the above exclusions or limitations may not be applicable in some states. This warranty provides specific legal rights; other rights that vary from state to state may also exist. This warranty shall not be applicable to the extent that any provisions of this warranty is prohibited by any Federal, State or Municipal Law that cannot be preempted.

Hardware Product Warranty Details

Warranty Period : ProSoft warranties hardware product for a period of one (1) year.

Warranty Procedure : Upon return of the hardware Product ProSoft will, at its option, repair or replace Product at no additional charge, freight prepaid, except as set forth below. Repair parts and replacement Product will be furnished on an exchange basis and will be either reconditioned or new. All replaced Product and parts become the property of ProSoft. If ProSoft determines that the Product is not under warranty, it will, at the Customer's option, repair the Product using current ProSoft standard rates for parts and labor, and return the Product freight collect.

B Product Specifications

The 3100/3150-MCM ("Modbus Communication Module") product family allows Allen-Bradley 1771 and 1746 I/O compatible processors to easily interface with other Modbus protocol compatible devices as either a Modus Master or a Modbus Slave.

The MCM product includes the following standard features:

General Specifications

- Two fully configurable serial ports, each capable of supporting Modbus Master or Modbus Slave. Available configurations include:

Port Configurations	Port 1	Port 2
Master-Master	Master	Master
Master-Slave	Master	Slave
Slave-Slave	Slave	Slave

- Support for the storage and transfer of up to 4000 registers to the PLC /SLC data tables
- Support movement of binary, integer, ASCII, and floating point data types
- Memory mapping is completely user definable through data table configuration
- RS-232C handshaking for SCADA radio/modem applications
- RS-422/RS-485 compatible for multidrop applications with up to 32 slaves per port
- Satellite and Packet Radio support with a configurable Inter-character Timeout available per port
- Software configuration (From processor ladder logic)

Slave Addr	:	1 to 247 (0 is broadcast)
Parity	:	None, odd, or even,
Stop Bit	:	1 or 2
Baud Rate	:	300 TO 38,400
RTS to TxD	:	0-65535 ms, 1 ms resolution
RTS Off	:	0-65535 ms, 1 ms resolution
Timeout	:	0-65535 ms, 1 ms resolution

- Response time
  The Modbus Master and Slave protocol drivers are written in Assembly and in a compiled higher level language. As such, the interrupt capabilities of the hardware are fully utilized to minimize delays, and to optimize the product's performance

Modbus Slave Specifications

- Protocol modes:
  RTU mode with CRC-16 error checking
  ASCII mode with LRC error checking (7 and 8 bit formats)
- Supported Modbus Function codes:

1	Read Output Status
2	Read Input Status
3	Read Multiple Data Registers
4	Read Input Registers
5	Force Single Coil (Latch/Unlatch)
6	Preset (Write) Single Data Register
8	Loopback Test (Test 0 only)
15	Multiple Coil Latch/Unlatch
16	Preset (Write) Multiple Data Register

- Supports broadcast commands from host
- Error Status and Communication Statistics returned to the ladder processor
- Pass Through Mode - configurable selection
  Select to transfer write commands from host directly to ladder for processing. Allows conditional acceptance of write data by ladder logic
- Command Routing Mode
  Supports Slave to Master routing for up to six slave addresses. Allows a supervisor on slave port to access data from the routed slaves

Modbus Master Specifications

- Protocol modes:
  RTU mode with CRC-16 error checking
  ASCII mode with LRC error checking (7 and 8 bit formats)

- Supported Modbus Function codes:

1	Read Output Status
2	Read Input Status
3	Read Multiple Data Registers
4	Read Input Registers
5	Force Single Coil (Latch/Unlatch)
6	Preset (Write) Single Data Register
15	Multiple Coil Latch/Unlatch
16	Preset (Write) Multiple Data Register

- Supports up to 100 Command List entries, each individually configurable with the following parameters:
  - Port/Mode Selection
  - Slave Address
  - Function Code
  - Source/Destination data address
  - Number of values to transfer
  - Polling Time
- Command Control Mode
  - Allows individual command execution control to be done in ladder logic enabling a list of commands to be executed based on events in the PLC/SLC
- Individual command 'Done' and 'Error' bits available
- Support for 'Event Driven' Writes initiated directly from ladder logic
- Individual Command Error Statue codes returned to the ladder processor
- Supports broadcast commands to slaves

Hardware Specifications

- Backplane Current Load :
  - 3100 : 0.65 A
  - 3150 : 0.15 A at 5 V
    0.04 A at 24 V
- Operating Temperature : 0 to 60 °C
- Storage Temperature : -40 to 85 °C
- Connections :
  - 3100 : 2 - DB25 Female Connectors
  - 3150 : 2 - DB9 Male Connectors

C Modbus Protocol Specification

Read Output Status (Function Code 01)

Query
This function allows the user to obtain the ON/OFF status of logic coils used to control discrete outputs from the addressed slave only. Broadcast mode is not supported with this function code. In addition to the slave address and function fields, the message requires that the information field contain the initial coil address to be read (Starting Address) and the number of locations that will be interrogated to obtain status data.

The addressing allows up to 2000 coils to be obtained at each request; however, the specific slave device may have restrictions that lower the maximum quantity. The coils are numbered from zero; (coil number 1 = zero, coil number 2 = one, coil number 3 = two, etc.).

Figure C1 is a sample read output status request to read coils 0020 to 0056 from slave device number 11.

ADR	FUNC	DATA START PT HO	DATA START PT LO	DATA # OF PTS HO	DATA # OF PTS LO	ERROR CHECK FIELD
11	01	00	13	00	25	CRC

Figure C1 Read Output Status Query Message

Response
An example response to Read Output Status is as shown in Figure C2. The data is packed one bit for each coil. The response includes the slave address, function code, quantity of data characters, the data characters, and error checking. Data will be packed with one bit for each coil (1 = ON, 0 = OFF). The low order bit of the first character contains the addressed coil, and the remainder follow. For coil quantities that are not even multiples of eight, the last characters will be filled in with zeros at high order end. The quantity of data characters is always specified as quantity of RTU characters, i.e., the number is the same whether RTU or ASCII is used.

Since the slave interface device is serviced at the end of a controller's scan, data will reflect coil status at the end of the scan. Some slaves will limit the quantity of coils provided each scan; thus, for large coil quantities, multiple PC transactions must be made using coil status from sequential scans.

ADR	FUNC	BYTE COUNT	DATA COIL STATUS 20-27	DATA COIL STATUS 28-35	DATA COIL STATUS 36-43	DATA COIL STATUS 44-51	DATA COIL STATUS 52-56	ERROR CHECK FIELD
11	01	05	CD	6B	B2	OE	1B	CRC

Figure C2 Read Output Status Response Message

The status of coils 20-27 is shown as CD(HEX) = 1100 1101 (Binary). Reading left to right, this shows that coils 27, 26, 23, 22, and 20 are all on. The other coil data bytes are decoded similarly. Due to the quantity of coil statuses requested, the last data field, which is shown 1B (HEX) = 0001 1011 (Binary), contains the status of only 5 coils (52-56) instead of 8 coils. The 3 left most bits are provided as zeros to fill the 8-bit format.

Read Input Status (Function Code 02)

Query
This function allows the user to obtain the ON/OFF status of discrete inputs in the addressed slave PC Broadcast mode is not supported with this function code. In addition to the slave address and function fields, the message requires that the information field contain the initial input address to be read (Starting Address) and the number of locations that will be interrogated to obtain status data.

The addressing allows up to 2000 inputs to be obtained at each request; however, the specific slave device may have restrictions that lower the maximum quantity. The inputs are numbered form zero; (input 10001 = zero, input 10002 = one, input 10003 = two, etc., for a 584).

Figure C3 is a sample read input status request to read inputs 10197 - 10218 from slave number 11.

ADR	FUNC	DATA START PT HO	DATA START PT LO	DATA #OF PTS HO	DATA #OF PTS LO	ERROR CHECK FIELD
11	02	00	C4	00	16	CRC

Figure C3 Read Input Status Query Message

Response

An example response to Read input status is as shown in Figure C4. The data is packed one bit for each input. The response includes the slave address, function code, quantity of data characters, the data characters, and error checking. Data will be packed with one bit for each input (1=ON, 0=OFF). The lower order bit of the first character contains the addressed input, and the remainder follow. For input quantities that are not even multiples of eight, the last characters will be filled in with zeros at high order end. The quantity of data characters is always specified as a quantity of RTU characters, i.e., the number is the same whether RTU or ASCII is used.

Since the slave interface device is serviced at the end of a controller's scan, data will reflect input status at the end of the scan. Some slaves will limit the quantity of inputs provided each scan; thus, for large coil quantities, multiple PC transactions must be made using coil status for sequential scans.

ADR	FUNC	BYTE COUNT	DATA DISCRETE INPUT 10197-10204	DATA DISCRETE INPUT 10205-10212	DATA DISCRETE INPUT 10213-10218	ERROR CHECK FIELD
11	02	03	AC	DB	35	CRC

Figure C4 Read Input Status Response Message

The status of inputs 10197-10204 is shown as AC (HEX) = 10101 1100 (binary). Reading left to right, this show that inputs 10204, 10202, and 10199 are all on. The other input data bytes are decoded similar.

Due to the quantity of input statuses requested, the last data field which is shown as 35 HEX = 0011 0101 (binary) contains the status of only 6 inputs (10213-102180) instead of 8 inputs. The two left-most bits are provided as zeros to fill the 8-bit format.

Read Holding Registers (Function Code 03)

Query

Read holding registers (03) allows the user to obtain the binary contents of holding registers 4xxxx in the addressed slave. The registers can store the numerical values of associated timers and counters which can be driven to external devices. The addressing allows up to 125 registers to obtained at each request; however, the specific slave device may have restriction that lower this maximum quantity. The registers are numbered form zero (40001 = zero, 40002 = one, etc.). The broadcast mode is not allowed.

The example below reads registers 40108 through 40110 from slave 584 number 11.

ADR	FUNC	DATA START REG HO	DATA START REG LO	DATA #OF REGS HO	DATA #OF REGS LO	ERROR CHECK FIELD
11	03	00	6B	00	03	CRC

Figure C5 Read Holding Register Query Message

Response

The addressed slave responds with its address and the function code, followed by the information field. The information field contains 1 byte describing the quantity of data bytes to be returned. The contents of the registers requested (DATA) are two bytes each, with the binary content right justified within each pair of characters. The first byte includes the high order bits and the second, the low order bits.

Since the slave interface device is normally serviced at the end of the controller's scan, the data will reflect the register content at the end of the scan. Some slaves will limit the quantity of register content provided each scan; thus for large register quantities, multiple transmissions will be made using register content from sequential scans.

In the example below, the registers 40108-40110 have the decimal contents 555, 0, and 100 respectively.

ADR	FUNC	BYTE CNT	H.O. DATA	L.O. DATA	H.O. DATA	L.O. DATA	H.O. DATA	L.O. DATA	ERROR CHECK FIELD
11	03	06	02	2B	00	00	00	64	CRC

Figure C6 Read Holding Register Response Message

Read Input Registers (Function Code 04)

Query

Function code 04 obtains the contents of the controller's input registers at addresses 3xxxx. These locations receive their values from devices connected to the I/O structure and can only be referenced, not altered from within the controller, The addressing allows up to 125 registers to be obtained at each request; however, the specific slave device may have restrictions that lower this maximum quantity. The registers are numbered for zero (30001 = zero, 30002 = one, etc.). Broadcast mode is not allowed.

The example below requests the contents of register 3009 in slave number 11.

ADR	FUNC	DATA START REG HO	DATA START REG LO	DATA #OF REGS HO	DATA #OF REGS LO	ERROR CHECK FIELD
11	04	00	08	00	01	CRC

Figure C7 Read Input Register Query Message

Response

The addressed slave responds with its address and the function code followed by the information field. The information field contains 1 byte describing the quantity of data bytes to be returned. The contents of the registers requested (DATA) are 2 bytes each, with the binary content right justified within each pair of characters. The first byte includes the high order bits and the second, the low order bits.

Since the slave interface is normally serviced at the end of the controller's scan, the data will reflect the register content at the end of the scan. Each PC will limit the quantity of register contents provided each scan; thus for large register quantities, multiple PC scans will be required, and the data provided will be form sequential scans.

In the example below the register 3009 contains the decimal value 0.

ADR	FUNC	BYTE COUNT	DATA INPUT REG HO	DATA INPUT REG LO	ERROR CHECK FIELD
11	04	02	00	00	E9

Figure C8 Read Input Register Response Message

Force Single Coil (Function Code 5)

Query

This message forces a single coil either ON or OFF. Any coil that exists within the controller can be forced to either state (ON or OFF). However, since the controller is actively scanning, unless the coil is disabled, the controller can also alter the state of the coil. Coils are numbered from zero (coil 0001 = zero, coil 0002 = one, etc.). The data value 65,280 (FF00 HEX) will set the coil ON and the value zero will turn it OFF; all other values are illegal and will not affect that coil.

The use of slave address 00 (Broadcast Mode) will force all attached slaves to modify the desired coil.

NOTE
Functions 5, 6, 15, and 16 are the only messages that will be recognized as valid for broadcast.

The example below is a request to slave number 11 to turn ON coil 0173.

ADR	FUNC	DATA COIL # HO	DATA COIL # LO	DATA ON/OFF IND	DATA	ERROR CHECK FIELD
11	05	00	AC	FF	00	CRC

Figure C9 Force Single Coil Query Message

Response

The normal response to the Command Request is to re-transmit the message as received after the coil state has be altered.

ADR	FUNC	DATA COIL # HO	DATA COIL # LO	DATA ON/ OFF	DATA	ERROR CHECK FIELD
11	05	00	AC	FF	00	CRC

Figure C10 Force Single Coil Response Message

The forcing of a coil via MODBUS function 5 will be accomplished regardless of whether the addressed coil is disabled or not. (*In ProSoft products, the coil is only affected if the necessary ladder logic is implemented).*

NOTE
The Modbus protocol does not include standard functions for testing or changing the DISABLE state of discrete inputs or outputs. Where applicable, this may be accomplished via device specific Program commands. (*In ProSoft products, this is only accomplished through ladder logic programming).*

Coils that are reprogrammed in the controller logic program are not automatically cleared upon power up. Thus, if such a coil is set ON by function Code 5 and (even months later), an output is connected to that coil, the output will be "hot".

Preset Single Register (Function Code 06)

Query

Function (06) allows the user to modify the contents of a holding register. Any holding register that exists within the controller can have its contents changed by this message. However, since the controller is actively scanning, it also can alter the content of any holding register at any time. The values are provided in binary up to the maximum capacity of the controller unused high order bits must be set to zero. When used with slave address zero (Broadcast mode) all slave controllers will load the specified register with the contents specified.

NOTE
Functions 5, 6, 15 and 16 are the only messages that will be recognized as valid for broadcast.

ADR	FUNC	DATA START REG HO	DATA START REG LO	DATA #OF REGS HO	DATA #OF REGS LO	ERROR CHECK FIELD
11	06	00	01	00	03	CRC

Figure C11 Preset Single Register Query Message

Response
The response to a preset single register request is to re-transmit the query message after the register has been altered.

ADR	FUNC	DATA REG HI	DATA REG LO	DATA INPUT REG HO	DATA INPUT REG LO	ERROR CHECK FIELD
11	06	00	01	00	03	CRC

Figure C12 Preset Single Register Response Message

Force Multiple Coils (Function Code 15)

Query
This message forces each coil in a consecutive block of coils to a desired ON or OFF state. Any coil that exists within the controller can be forced to either state (ON or OFF). However, since the controller is actively scanning, unless the coils are disabled, the controller can also alter the state of the coil. Coils are numbered from zero (coil 00001 = zero, coil 00002 = one, etc.). The desired status of each coil is packed in the data field, one bit for each coil (1= ON, 0= OFF). The use of slave address 0 (Broadcast Mode) will force all attached slaves to modify the desired coils.

NOTE
Functions 5, 6, 15, and 16 are the only messages (other than Loopback Diagnostic Test) that will be recognized as valid for broadcast.

The following example forces 10 coils starting at address 20 (13 HEX). The two data fields, CD =1100 and 00 = 0000 000, indicate that coils 27, 26, 23, 22, and 20 are to be forced on.

ADR	FUNC	H.O. ADD	L.O. ADD	QUANTITY		BYTE CNT	DATA COIL STATUS 20-27	DATA COIL STATUS 28-29	ERROR CHECK FIELD
11	0F	00	13	00	0A	02	CD	00	CRC

Figure C15 Force Multiple Coils Query Message

Response
The normal response will be an echo of the slave address, function code, starting address, and quantity of coils forced.

ADR	FUNC	H.O. ADDR	L.O. ADDR	QUANTITY		ERROR CHECK FIELD
11	0F	00	13	00	0A	CRC

Figure C16 Force Multiple Coils Response Message

The writing of coils via Modbus function 15 will be accomplished regardless of whether the addressed coils are disabled or not.

Coils that are unprogrammed in the controller logic program are not automatically cleared upon power up. Thus, if such a coil is set ON by function code 15 and (even months later) an output is connected to that coil, the output will be hot.

Preset Multiple Registers (Function Code 16)

Query

Holding registers existing within the controller can have their contents changed by this message (a maximum of 60 registers). However, since the controller is actively scanning, it also can alter the content of any holding register at any time. the values are provided in binary up to the maximum capacity of the controller (16-bit for the 184/384 and 584); unused high order bits must be set to zero. When specified registers with contents specified.

NOTE

Function codes 5, 6, 15, and 16 are the only messages that will be recognized as valid for broadcast.

ADR	FUNC	H.O. ADD	L.O. ADD	QUANTITY		BYTE CNT	H.O. DATA	L.O. DATA	H.O. DATA	L.O. DATA	ERROR CHECK FIELD
11	10	00	87	00	02	04	00	0A	01	02	CRC

Figure C13 Preset Multiple Coils Query Message

Response

The normal response to a function 16 query is to echo the address, function code, starting address and number of registers to be loaded.

ADR	FUNC	H.O. ADDR	L.O. ADDR	QUANTITY		ERROR CHECK FIELD
11	10	00	87	00	02	56

Figure C14 Preset Multiple Registers Response Message

D Jumper Configurations

Hardware Overview

When purchasing the MCM product, there are two available configurations. These choices are as follows:

		ProSoft Cat NumDescription
PLC		**SLC**
Module provided by ProSoft	3100	3150

When purchasing the module from ProSoft Technology, the jumper configurations will have been factory set to default positions for testing prior to shipment..

Module Jumper Configurations

The following section details the available jumper configurations for the 1771 and 1746 platform solutions. As needed, differences between the module based solutions and the firmware based solutions are highlighted.

3100 for the 1771 Platform

Following are the jumper positions for the ProSoft Technology 3100-MCM module:

Jumper	3100
JW1	N/A
JW2	N/A
JW3	N/A
JW4	Flash Pgm/Run Mode
JW5	8 Pt
JW6	Not Used
JW7	Enabled
JW8	Port 2 RS232/422/485 config
JW9	Port 1 RS232/422/485 config

JW4 Flash Pgm/Run Mode Select — Run Position

The position of this jumper should only be changed if needing to reprogram the MCM FLASH memory. This will only need to be done if the module is to be upgraded in the field to a later version of firmware.

JW5 Backplane 8/16 point — 8 Point

The module should be operated in the 8 point configuration unless specifically directed otherwise by the factory.

JW7 Battery Enable / Disable — Enabled

This jumper should be placed in the Enabled position when the module is powered up. Although not critical to the operation of the module, this will back up some data registers in the module during a power failure or reset.

JW8/9 RS Configuration for Port 1 and 2 — RS-232,422,485

The default from factory is RS-232, but all options are supported by the MCM firmware

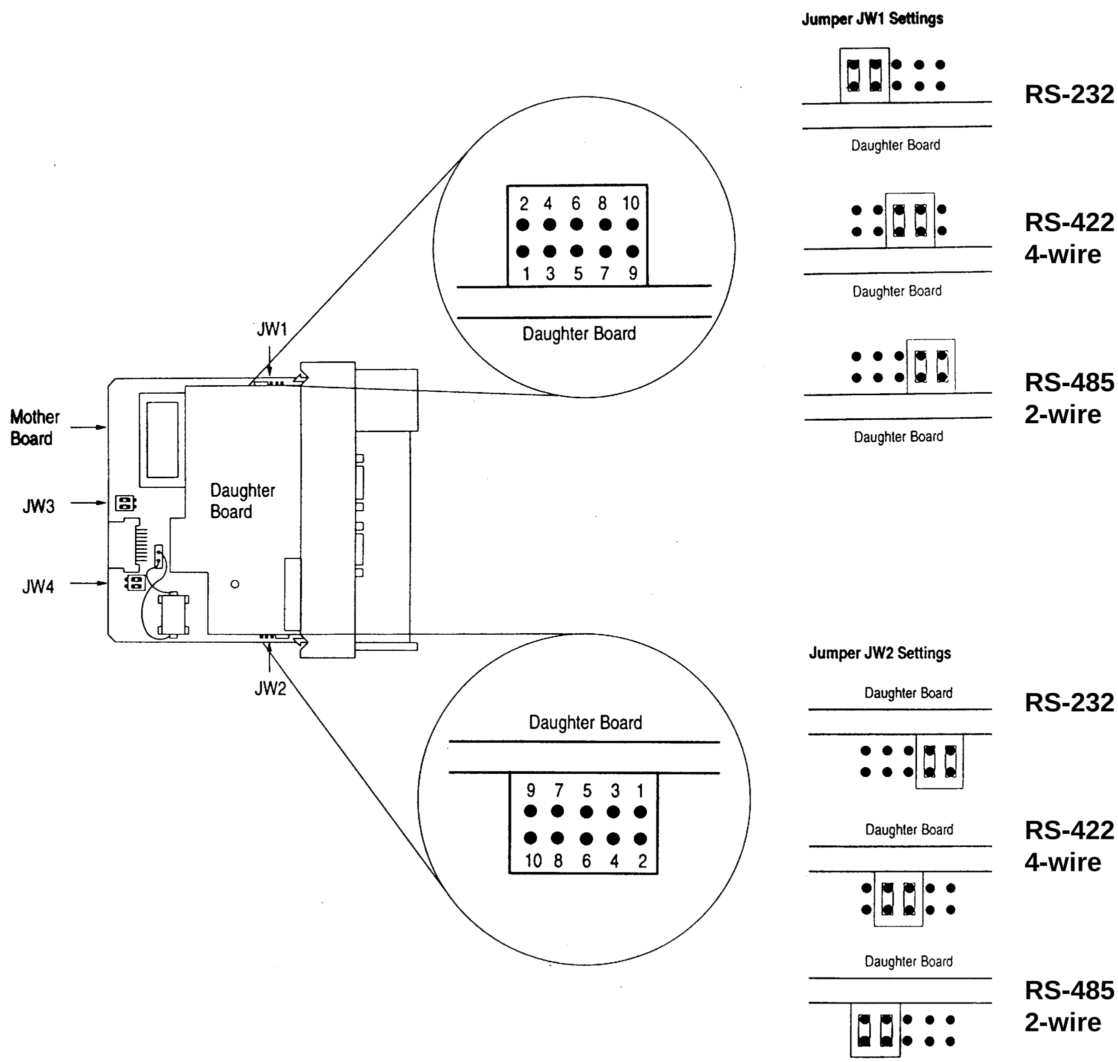
3150 for the 1746 Platform

Following are the jumper positions for the ProSoft Technology 3150-MCM module :

Jumper	3150-MCM
JW1	As Needed
JW2	As Needed
JW3	N/A
JW4	N/A

JW1/2 RS configuration for port 1 and 2 — RS-232 Position

The default from factory is RS-232, but RS-422 and RS-485 are supported by the firmware and hardware. See the following diagram:


Jumper JW1 Settings
RS-232
Daughter Board
RS-422
4-wire
Daughter Board
RS-485
2-wire
Daughter Board
2 4 6 8 10
1 3 5 7 9
Daughter Board
JW1
Mother Board
JW3
JW4
Daughter Board
JW2
Daughter Board
9 7 5 3 1
10 8 6 4 2
Jumper JW2 Settings
Daughter Board
RS-232
Daughter Board
RS-422
4-wire
Daughter Board
RS-485
2-wire

E Product Revision History

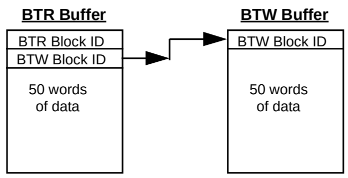
05/24/95	Revision 1.40 - 2 - Initial release of product
07/04/95	Revision 1.41 - 3 - Fix for FC 15 when writing less than 8 bits. - Add additional support for looped operation on ports
07/23/95	Revision 1.42 - 4 - Fix bug causing both ports to shut down when in multi-master mode
09/01/96	Revision 1.44 - Change RTS delays to support word values instead of just byte values. Now able to range RTS delays up to 65534. - Re-link of 1.42 to support the 1771-DB firmware upgrade - Add event driven write capability for master port - Add pass-through mode for slave port - Support floating point variable type (FC 3, 6, 16 greater than register 47001) - Incorporate better noise immunity logic for beginning of messages - Add support for > 255 characters to buffer to support full ASCII mode needs - Add master-slave routing capabilities for up to 6 slave addresses - Add Read Block Start and Write Block Start configuration values - In Master Mode, only return Error 8 after have retried 3 times
12/31/96	Revision 2.0 Released

F Read, Write and Command Block Count Values usage

As part of the configuration process, the User is able to configure several parameters in the Communication Configuration Data Block which have a strong impact on how the module transfers data with the PLC/SLC processor.

Overview

As shown in Section 4 and 5 of the manual the BTR buffer contains the BTR and the BTW Block ID numbers. The BTR Block ID is used to identify the data contents, while the BTW Block ID is used by the ladder logic to determine which data to move to the module. Diagramatically, the relationship is as follows:



Configuration Parameters

Three parameters which are important to the transfer of data are:

Read Block Count : This value represents the number of 50 word data blocks which are to be transferred from the MCM Module to the processor. The blocks returned from the module start at the value entered in the Read_Block_Start register and increments from there

Write Block Count : This value represents the number of 50 word data blocks which are to be transferred from the processor to the MCM Module.

Command Block Count : This value represents the number of 50 word Command Blocks which are to be transferred from the processor to the MCM Module.

These values are used by the module in order to determine how the BTW and BTR Block ID Codes are to be manipulated. Part of the functionality that the module provides is to control the incrementing and resetting of the BTR and BTW Block ID codes. This was done in the interest of limiting the amount of ladder logic required to support the module.

Module Operation

As are result of the configuration parameters entered, the module will cycle through the range of BTW and BTR Blocks. The cycle is based on the following equations:

BTW Block ID

```
        if ( BTW Block ID  >= Write_Block_Cnt ) then BTW Block ID = 80
        elseif( BTW Block ID  >= 80 + Command_Block_Cnt) then BTW Block ID  = Write_Block_Start
    else BTW block ID = BTW block ID + 1
```

BTR Block ID

```
        if ( BTR Block ID >= Read_Block_Cnt ) then BTR Block ID = Read_Block_Start
        else BTR block ID = BTR block ID + 1
```

As an example, assume that we are configured with the following values:

Read_Block_Cnt	4
Write_Block_Cnt	1
Command_Block_Cnt	2
Read_Block_Start	1
Write_Block_Start	0

These configuration values would lead to the following cycle of Block ID codes:

BTW Block ID	BTR Block ID
0	1
80	2
81	3
0	4
80	1
81	2
0	3

Note that there is no implicit relationship between the absolute value in the BTW and the BTR Block ID.

G Example Ladder Logic

The following example logic has been provided to assist you in developing applications more effectively. These examples are provided under a separate manual entitled Example Ladder Logic (Two such manuals are available, one for the PLC and one for SLC examples)

Slave Mode Examples

Example #1 : Slave Mode w/ Pass-Thru - Minimum Configuration

MCM5EX1S	**PLC 5**
MCM3EX1S	**SLC 5/03**

Example #2 : Slave Mode w/ Pass-Thru - Expanded Application

MCM5EX2S	**PLC 5**
MCM3EX2S	**SLC 5/03**

Master Mode Examples

Example #1 : Master Mode - Basic Application

MCM5EX1M	**PLC 5**
MCM3EX1M	**SLC 5/03**

Example #2 : Master Mode w/ Command Control

MCM5EX2M	**PLC 5**
MCM3EX2M	**SLC 5/03**